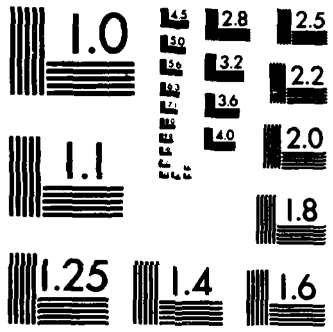
AD-A137 084   SOFTWARE ACQUISITION RESOURCE EXPENDITURE (SARE) DATA   1/2
COLLECTION METHODOLOGY(U) MITRE CORP BEDFORD MA
R L DUMAS DEC 83 MTR-9031 ESD-TR-83-214
UNCLASSIFIED   F19628-82-C-0001   F/G 9/2   NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

12

ESD-TR-83-214 MTR-9031

FINAL REPORT: SOFTWARE ACQUISITION RESOURCE EXPENDITURE (SARE) DATA COLLECTION METHODOLOGY

By
R. L. DUMAS

DECEMBER 1983

Prepared for
COMPTROLLER, COST ESTIMATING AND ANALYSIS DIVISION
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Massachusetts

Project No. 6810
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-82-C-0001

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

[illegible]

JOSEPH P. DEAN, Capt, USAF
SARE Project Officer

[illegible]

RONALD S. BOWEN, Lt Col, USAF
Director of Cost Analysis
Comptroller

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): MTR-9031 ESD-TR-83-214	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: The MITRE Corporation; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Burlington Road, Bedford, MA 01730	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Comptroller, Cost Estimating & Analysis Division; 8b. OFFICE SYMBOL (If applicable): ACCE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F19628-82-C-0001
8c. ADDRESS (City, State and ZIP Code): Electronic Systems Division, Hanscom AFB, MA 01731	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
	6810		

11. TITLE (Include Security Classification): FINAL REPORT: SOFTWARE ACQUISITION (over)

12. PERSONAL AUTHOR(S): R. L. Dumas

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final Report	FROM ______ TO ______	1983 December	136

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	SARE METHODOLOGY
			SOFTWARE COST ESTIMATION
			WORK BREAKDOWN STRUCTURE

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

One of the major shortfalls in software cost estimation is the lack of a well-defined, well-structured database. This report is the culmination of a multi-year effort to develop contractual documents to effect quality software data collection on defense programs. It includes as attachments a proposed draft military standard for software work breakdown structures and a data item description for reporting project attributes that impact software cost and schedule. The evolution of the documents is discussed and procedures for implementing data collection on defense programs are provided. Readers are asked to comment on the SARE methodology using a questionnaire provided as the third attachment.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☒ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Susan R. Gilbert	22b. TELEPHONE NUMBER (Include Area Code): (617) 271-8088; 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

11. (Concluded)

RESOURCE EXPENDITURE (SARE) DATA COLLECTION METHODOLOGY

ACKNOWLEDGEMENTS

The SARE methodology has taken on several forms since its inception. During that time many individuals have contributed to its development. The original research was performed by William E. Byrne, Donna M. Cosgrove, Manfred P. Friedman, John B. Glore, Shashi Phoha, and William M. Stein, all of the MITRE Corporation, under the leadership of John B. Glore. Through most of the SARE development, guidance and direction were provided by Maj. Joseph A. Duquette, then of AFSC ESD/ACCI and presently of USAF/ACMC. More recently guidance has been provided by Capt. Wesley W. Jones, ESD/ACCI, and Capt. Joseph P. Dean, ESD/ACCE.

William E. Byrne and Roger L. Dumas were responsible for intermediate versions of the SARE documents with guidance from Robert L. Hamilton. The present version of the SARE methodology presented herein was prepared by Roger L. Dumas with contributions from Capt. Joseph P. Dean, ESD/ACCE, Charles S. Enright, Peter Frasca, Mary Jean Hayes, Robert G. Howe, John H. James, and William M. Stein. Many others, too numerous to list, have reviewed drafts of the documents over the past several years and provided valuable comments.

This document has been prepared by The MITRE Corporation under Project 6810, Contract F19628-82-C-0001. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

A-1

PREFACE

MIL-STD-X, dated July 1983, shown as Attachment 1, and Data Item Description, Attachment 2, have two sets of page numbers. They may be pulled out and used as stand-alone documents.

TABLE OF CONTENTS

TABLE OF CONTENTS (concluded)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

OVERVIEW OF SARE REPORTING

FRAMING THE PROBLEM

The Department of Defense (DoD) continues to experience difficulty estimating the cost of software development on defense system acquisition programs. Factors contributing to cost estimate uncertainty are classified into three categories: project-related, technology-related, and deficiencies in the science of software cost estimation.

Project-related factors include:

- o Poor requirements definition - the inability to state specific functional and performance requirements of a software system in an unambiguous way.
- o Requirements instability - the inability to pin down functional and performance requirements early in a program.

The technology-related factor has to do with the relative immaturity of software technology and its perception as an art form rather than a science. More specifically:

- o There exist no structured analysis technique and program design methodology that will together produce a well-defined, well-structured statement of functional requirement and translate that statement into a unique system design, regardless of the analyst or designer.

As a result, the configuration of a hardware/software system designed to meet a specific functional requirement can take on many varied forms, with a wide range of potential costs.

Project and technology factors aside, a final source of software cost estimation uncertainty has to do with the available software cost models and the databases on which their estimates are based.

Over the past 20 years, the number of software cost models has increased significantly (Nelson 1966, Aron 1969, Tecolote 1974, Wolverton 1974, Putnam 1978, Herd at al. 1977, Freiman and Park 1979, Walston and Felix 1977, Black et al. 1977, Bourdon and Duquette 1978, Boehm 1981). Early models tended to be poorly defined and based on poorly defined data. The definition of the models themselves has improved in recent years. However, by and large, the databases on which they are based continue to be suspect. There is and always will be a need to collect data on software projects to validate and calibrate cost models as software technology continues to evolve.

There are presently a limited number of databases available to support software cost estimation research. Several early models were based on data from 169 projects compiled by the Systems Development Corporation (Fleishman 1966). Although the SDC data included an uncommonly large number of factors affecting software development, the data was from relatively small projects (most less than 10K lines of code), all of which were completed p[illegible]r to 1966.

The primary source of data for several of the mo[illegible] recent models has been the Data and Analysis Center for Softw[illegible] (DACS) located at Griffiss Air Force Base, New York. The DAC[illegible]aintains several databases (DACS 1982), the principal being the [illegible]ductivity Dataset of roughly 400 projects. The major shortfall[illegible] ne Productivity Dataset are the limited number of factors [illegible]counted for and poor definition. Factors such as "project size" and "project effort" mean different things for different projects in the dataset.

Several other cost models are based on proprietary or unpublished databases. A recent model, the Constructive Cost Model (COCOMO) (Boehm 1981), has a published database of 63 projects for which "considerable effort has been devoted to ensuring that the data in the COCOMO database is consistent with respect to cost driver attribute ratings, and the definitions of such quantities as development, man-month, project, and (delivered source instructions) agree with the COCOMO assumptions." COCOMO however, is the exception.

The SARE data collection methodology addresses this final software cost estimation deficiency, specifically, the lack of a quality database of software development costs and project attributes for defense system acquisition programs. The SARE development has been directed by the Electronic Systems Division (ESD) of AFSC with funding from AFSC and other branches of DoD. The methodology is consistent with standard military procurement practices and can be applied to all major DoD programs.

ORGANIZATION OF THIS REPORT

The remainder of this section provides background to help the reader understand the context in which SARE data collection is conducted. It describes the features of the defense system acquisition environment that impact software cost data collection, the SARE concept of operation, and a summary of the evolution of the SARE methodology.

Sections 2 and 3 provide overviews of the two documents used to implement SARE data collection: a draft military standard for software work breakdown structures (WBSs) and a draft data item description (DID) for reporting software attributes on defense programs. The documents themselves appear in Attachments 1 and 2 of this report.

Section 4 discusses ESD's plans to conduct an industry/government review of the SARE documents and summarizes the results of a limited, preliminary review conducted during the Spring of 1983. Readers are asked in Section 4 to provide comments to ESD/ACC regarding implementation of the SARE methodology. An evaluation questionnaire is provided in Attachment 3 to assist reviewers. Section 5 states the major conclusions of the SARE development effort and provides recommendations for following up the effort.

SOFWARE DATA COLLECTION IN THE DEFENSE ENVIRONMENT

To understand the operation of the SARE methodology, one must first have a rudimentary understanding of the managerial and technical aspects of the environment in which it operates. The key DoD directives, instructions, and MIL-STDs which frame that environment are listed in Figure 1-1.

Any system that attempts to collect resource expenditure data (that is, dollars and manhours) must be integrated with government and contractor management systems. This, in turn, implies consistency with the cost/schedule control systems criteria (C/SCSC), delineated in DoDI 7000.2, "Performance Measurement on Selected Acquisition Programs."

The C/SCSC are a set of characteristics a contractor's management system must possess to assure the government that the contractor is capable of planning and controlling costs and schedule during the system development. There are 35 criteria arranged in the five categories summarized below:

KEY DOCUMENTS IMPACTING SOFTWARE DATA COLLECTION

FINANCIAL

DoDD 5000.1	Major System Acquisitions
DoDD 7000.1	Resource Management Systems of the Department of Defense
DoDI 7000.2	Performance Measurement on Selected Acquisitions
DoDI 7000.10	Contract Cost Performance, Funds Status and Cost/Schedule Status Reports
DoDI 7000.11	Contractor Cost Data Reporting
MIL-STD-881A	Work Breakdown Structures for Defense Materiel Items
AFSCP 173-5	Cost/Schedule Control Systems Criteria Joint Implementation Guide
AFSCP 173-6	Cost/Schedule Control System Criteria Joint Surveillance Guide

TECHNICAL

MIL-STD-483	Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs
MIL-STD-490	Specification Practices
MIL-STD-1521A	Technical Reviews and Audits for Systems, Equipment, Munitions and Computer Programs
MIL-STD-1679	Weapon System Software Development (Navy)

Figure 1-1. Key Documents Impacting Software Data Collection

1. Organization - define contractual effort and assign responsibilities for the work (5 criteria).

2. Planning and Budgeting - plan, schedule, budget, and authorize the work (11 criteria).

3. Accounting - accumulate costs of work and materials (7 criteria).

4. Analysis - compare planned and actual costs and analyze variances (6 criteria).

5. Revisions and Access to Data - incorporate changes and develop estimates of final costs; allow the government access to internal data to verify conformance (6 criteria).

AFLCP/AFSCP 173-5, "C/SCSC Joint Implementation Guide," discusses the criteria in depth and describes the process by which a contractor's cost/schedule control system is validated.

The first set of criteria, Organization, requires the contractor "define all authorized work and related resources to meet the requirements of the contract, using the framework of the CWBS."

The C/SCSC defer specific CWBS requirements to MIL-STD-881A, "Work Breakdown Structures for Defense Materiel Items." MIL-STD-881A, in turn, defines a WBS to be "a product-oriented family tree composed of hardware, services and data that result from project engineering efforts during the development and production of a defense materiel item, and that completely defines the project/program." (Note the absence of "software" in the WBS definition.) A contract WBS (CWBS) is simply a WBS applied to a particular contract.

The first three levels of a typical MIL-STD-881A CWBS for a defense system are presented in Table 1-1. This level of CWBS is normally mandated on the contractor by the government. The contractor then extends the CWBS to lower levels during the program, in accordance with MIL-STD-881A, to reflect how the work will be performed.

The Planning and Budgeting criteria require the contractor plan, schedule, and budget all authorized work in cost accounts that are located beneath the lowest level CWBS elements. A cost account is a managerial control point at which actual cost is accumulated and compared to budgeted cost. Cost accounts are further broken

TABLE 1-1

MIL-STD-881A WORK BREAKDOWN STRUCTURE

Level	Element
1	Defense System
2	Prime Mission Equipment
3	Integration and Assembly
3	Hardware Subsystem or End Item
.	.
.	.
.	.
3	Hardware Subsystem or End Item
3	Auxiliary Equipment
2	Training
3	Equipment
3	Services
3	Facilities
2	Peculiar Support Equipment
3	Organizational/Intermediate
3	Depot
2	Systems Test and Evaluation
3	Development Test and Evaluation
3	Operational Test and Evaluation
3	Mockups
3	Test and Evaluation Support
3	Test Facilities
2	System/Program Management
3	Systems Engineering
3	Project Management
2	Data
3	Technical Publications
3	Engineering Data
3	Management Data
3	Support Data
3	Data Depository
2	Operational/Site Activation
3	Contractor Technical Support
3	Site Construction
3	Site/Ship/Vehicle Conversion
3	System Assembly, Installation and Checkout on Site
2	Common Support Equipment
3	Organizational/Intermediate
3	Depot

TABLE 1-1 (Concluded)

Level	Element
2	Industrial Facilities
3	Construction/Conversion/Expansion
3	Equipment Acquisition or Modernization
3	Maintenance
2	Initial Spares and Initial Repair Parts

down into short-term work packages (normally of one to two months duration) to schedule work in detail and measure the "earned value" of work performed. The contractor accumulates financial data at the cost account level, summarizes the information up through higher levels of the CWBS, and reports actual versus budgeted expenditures in monthly Cost Performance Reports (DI-F-6000C). Figure 1-2 depicts the relationships among the CWBS, contractor organization structure, cost accounts and work packages.

To summarize the discussion thus far, consistent software cost data collection must be accomplished through a standard WBS. MIL-STD-881A is the authority governing WBS development but is presently deficient in the software area. This has spurred the creation of a draft software WBS military standard as part of the SARE methodology.

The C/SCSC deal with the management side of system acquisitions but stay away from technical details. The primary documents dealing with the technical side of software development are MIL-STD-483, "Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs;" MIL-STD-490, "Specification Practices;" and MIL-STD-1521A, "Technical Reviews and Audits for Systems, Equipment, Munitions and Computer Programs."

Together these documents define the principal software products, establish procedures for tracking and controlling changes, and establish program milestones at which intermediate products can be evaluated to measure technical progress.

Like MIL-STD-881A, these standards had their roots in hardware-oriented acquisitions and have been minimally updated in recent years to take into consideration aspects of the acquisition process which are peculiar to software. MIL-STD-1679, "Weapon System Software Development," is a Navy sponsored standard that attempts to fill the voids left by the others. Unfortunately, it differs significantly in its terminology and approach to the acquisition process. A new military standard, MIL-STD-SDS, "Proposed Military Standard on Defense System Software Development," has been proposed by the Joint Logistics Commanders (JLC). MIL-STD-SDS attempts to address the same needs as MIL-STD-1679 but using terminology and a model of the acquisition process similar to MIL-STD-483, MIL-STD-490, and MIL-STD-1521A.

At this writing, MIL-STD-SDS remains a draft. The SARE documents presented in this report reflect the acquisition process presently defined by MIL-STD-483, MIL-STD-490, and MIL-STD-1521A. Revisions needed to make the SARE documents consistent with

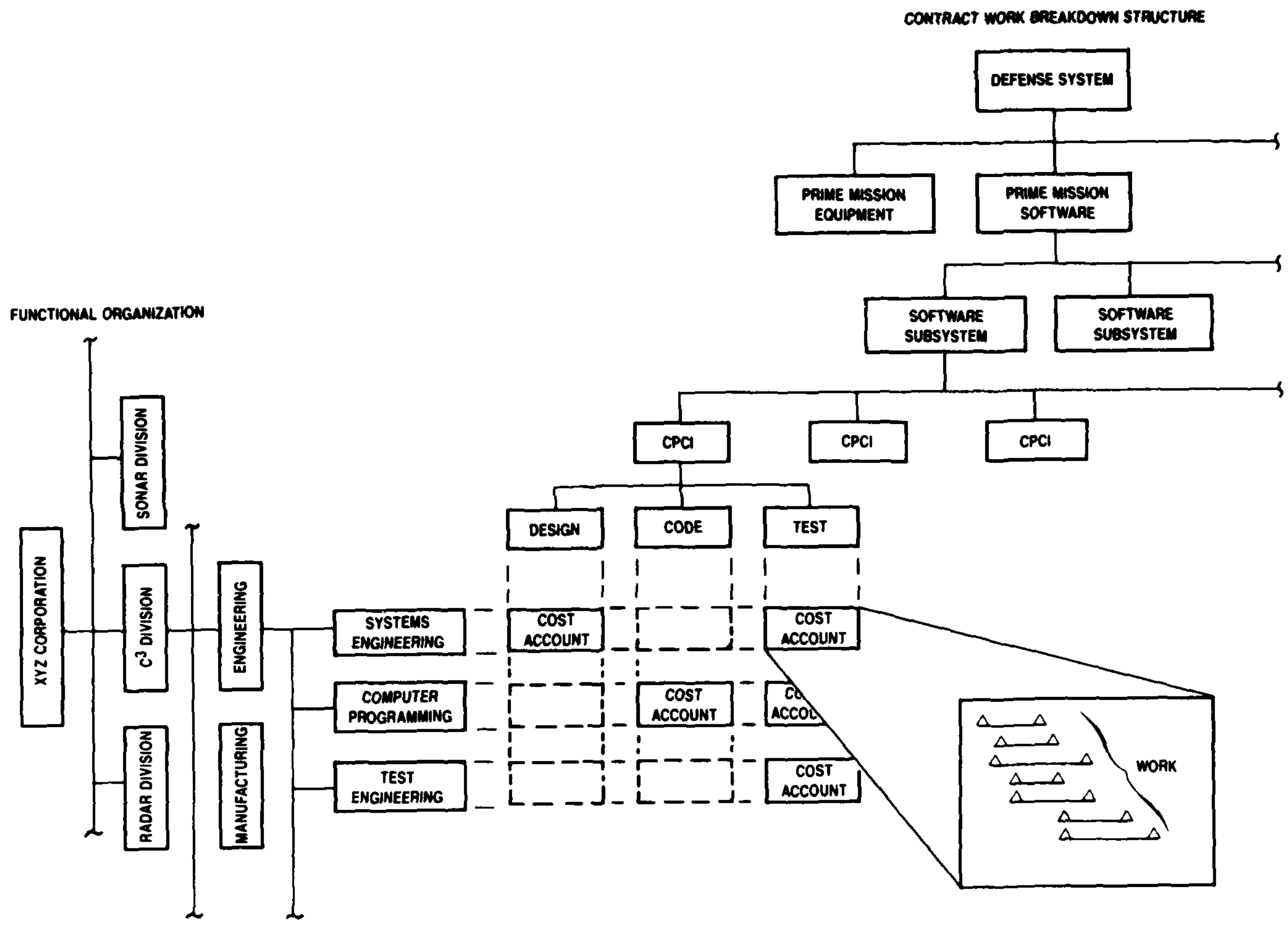


Figure 1-2. Work Packages and Cost Accounts in Relation to the CWBS

MIL-STD-SDS primarily involve terminology and can be made if and when MIL-STD-SDS is approved.

In summary, the SARE data collection methodology must balance the program management requirements of the C/SCSC and MIL-STD-881A with the technical products and program milestones defined by MIL-STD-483, MIL-STD-490, and MIL-STD-1521A.

SARE CONCEPT OF OPERATION

Software cost models take in attributes of software projects, relate the attributes to man-hours and cost through relationships derived from past experience, and put out cost and schedule estimates. Thus, the SARE methodology has to collect cost (that is, dollars and man-hours) and schedule data on the one hand, and technical characteristics to correlate to cost on the other hand. This must be done within the environment described in the previous section.

The first requirement of the SARE methodology is to establish well-defined, software-related CWBS elements for consistent cost data collection across programs. The form taken for the SARE CWBS elements is a draft military standard presented as Attachment 1 and discussed in Section 2. Draft MIL-STD-X, as it is referred to, is entitled "Draft Military Standard: Software Work Breakdown Structures for Defense System Acquisitions."

Draft MIL-STD-X extends the CWBS requirements of MIL-STD-881A in the area of software by providing instructions to the government agency and contractor for the identification of software components in the CWBS and the lower level extension of the CWBS with software products and services.

The second requirement of the SARE methodology is to provide a medium for data collection. The mechanisms for reporting data on defense programs are data item descriptions (DIDs) referenced in the contract data requirements list (CDRL) of the contract. A DID defines a reporting requirement that remains relatively invariant from program to program, that is, the data items to be reported and the reporting format. The CDRL specifies conditions for delivery, such as delivery dates and the distribution list. The CDRL also specifies any special tailoring instructions for the DIDs relative to the particular program.

The draft DID developed to effect SARE data collection is presented as Attachment 2 and discussed in Section 3.

Figure 1-3 demonstrates how the SARE documents are placed on contract. When a request for proposal (RFP) package is released in a solicitation or invitation for bids, it contains among other things, a statement of work (SOW) and the CDRL. The SOW states the contractor tasks needed to develop the defense system. One section of the SOW addresses the contractor's program management system, including adherence to the C/SCSC and MIL-STD-881A.

To invoke the SARE MIL-STD-X, a sentence must be added to the CWBS paragraph of the SOW. A model SOW paragraph used to establish CWBS requirements, including adherence to Draft MIL-STD-X, is provided below:

> <u>Contract Work Breakdown Structure (CWBS)</u>
>
> The contractor shall maintain the CWBS and dictionary in compliance with MIL-STD-881A. The contractor shall extend the CWBS to lower levels to reflect how the work is performed. The extension shall include all configuration items. The CWBS shall also be extended to account for software development activities in accordance with MIL-STD-X. The contractor shall use the CWBS as the primary framework for planning, budgeting, and reporting cost and schedule status to the government. The contractor shall update the CWBS as additional system definition is accomplished and may propose alternatives for improvement. Changes to the CWBS or associated definitions require prior approval of the government. The contractor shall deliver the CWBS in accordance with the CDRL.

The SOW also instructs the contractor to report SARE data during the program. An entry similar to the following would appear in the program management section of the SOW:

> <u>Software Acquisition Resource Expenditure (SARE) Reporting</u>
>
> The contractor shall contribute to the SARE database by providing data on the software development effort in accordance with the CDRL. The contractor shall designate a focal point responsible for SARE reporting. The source of the cost and man-hour data shall be cost accounts established in accordance with DoDI 7000.2. The source of the technical data shall be the technical personnel responsible for the software development.

Finally, the CDRL must include an entry specifying delivery requirements for the SARE DID. Figure 1-4 provides a model CDRL entry that establishes the preferred level of SARE reporting. The content of each block in the CDRL entry is defined in the DD

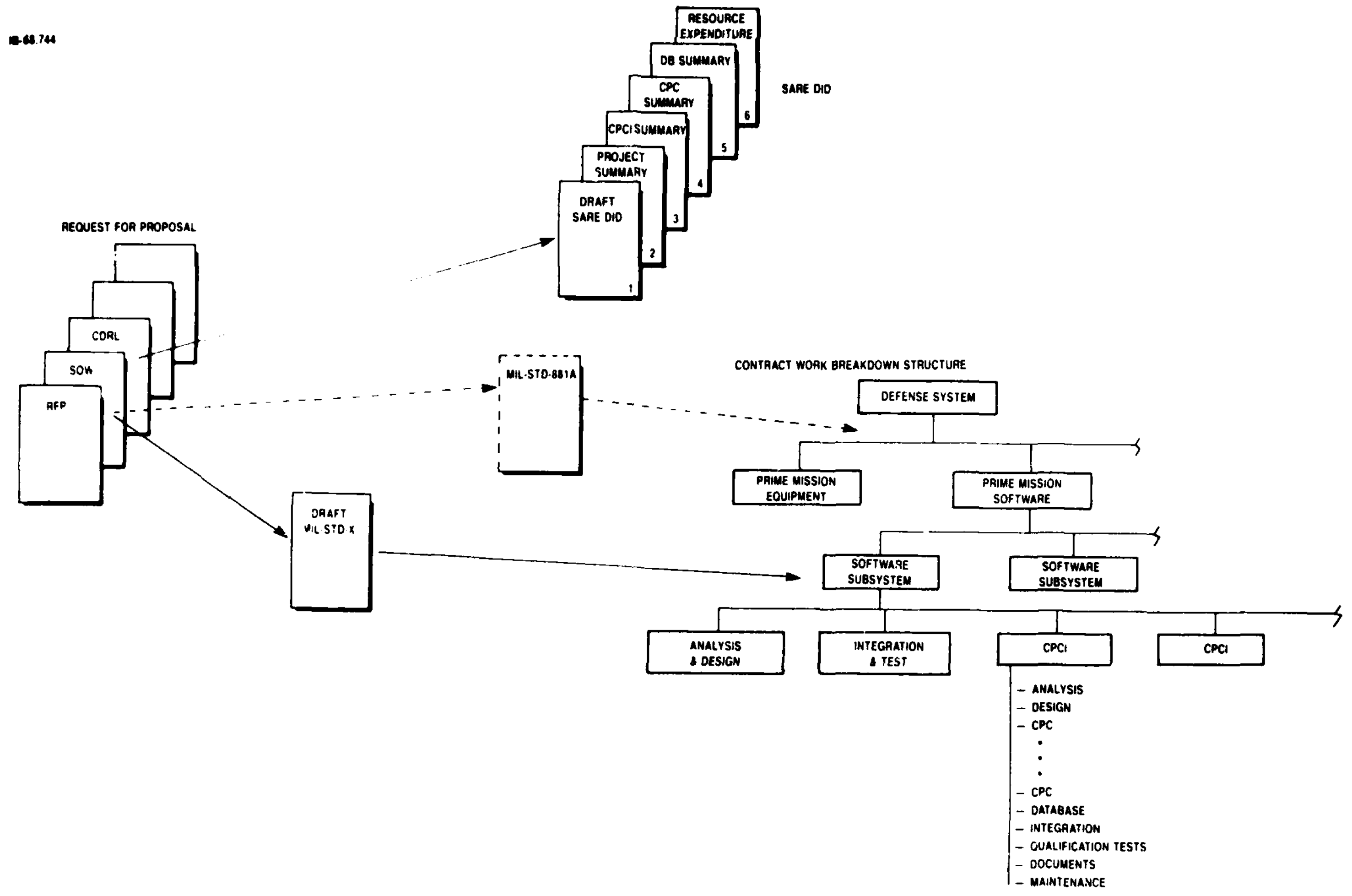
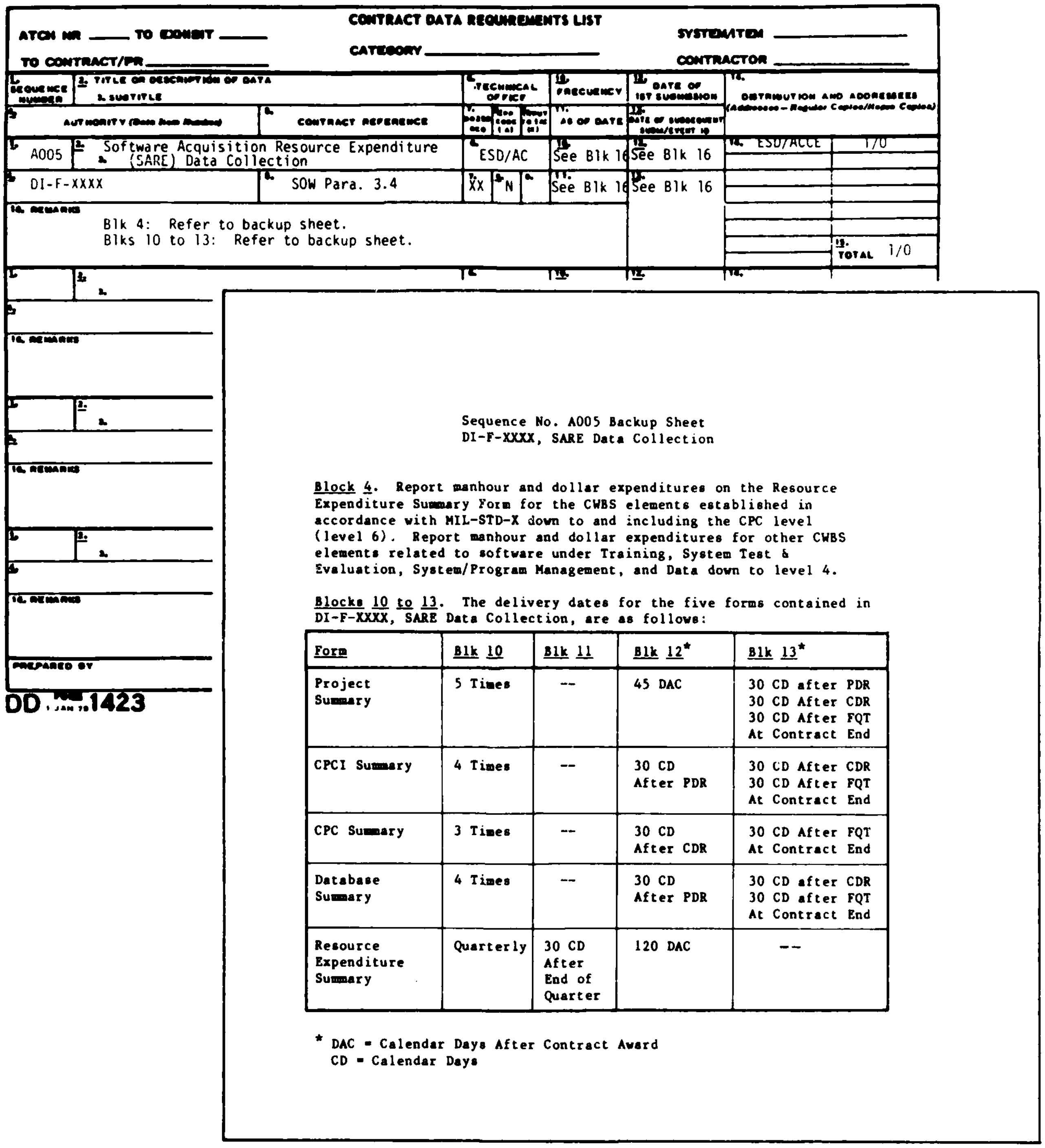


Figure 1-3. SARE Implementation



CONTRACT DATA REQUIREMENTS LIST

ATCH NR ____ TO EXHIBIT ____ SYSTEM/ITEM ____

CATEGORY ____

TO CONTRACT/PR ____ CONTRACTOR ____

1. SEQUENCE NUMBER	2. TITLE OR DESCRIPTION OF DATA / 3. SUBTITLE	4. TECHNICAL OFFICE	10. FREQUENCY	12. DATE OF 1ST SUBMISSION	14. DISTRIBUTION AND ADDRESSEES
A005	Software Acquisition Resource Expenditure (SARE) Data Collection	ESD/AC	See Blk 16	See Blk 16	ESD/ACCE 1/0

5. AUTHORITY	6. CONTRACT REFERENCE	7.	8.	9.	11. AS OF DATE	13. DATE OF SUBSEQUENT SUBM/EVENT ID
DI-F-XXXX	SOW Para. 3.4	XX	N		See Blk 16	See Blk 16

16. REMARKS

Blk 4: Refer to backup sheet.
Blks 10 to 13: Refer to backup sheet.

15. TOTAL 1/0

16. REMARKS

16. REMARKS

16. REMARKS

PREPARED BY

DD FORM 1 JAN 1423

Sequence No. A005 Backup Sheet
DI-F-XXXX, SARE Data Collection

Block 4. Report manhour and dollar expenditures on the Resource Expenditure Summary Form for the CWBS elements established in accordance with MIL-STD-X down to and including the CPC level (level 6). Report manhour and dollar expenditures for other CWBS elements related to software under Training, System Test & Evaluation, System/Program Management, and Data down to level 4.

Blocks 10 to 13. The delivery dates for the five forms contained in DI-F-XXXX, SARE Data Collection, are as follows:

Form	Blk 10	Blk 11	Blk 12*	Blk 13*
Project Summary	5 Times	--	45 DAC	30 CD after PDR 30 CD After CDR 30 CD After FQT At Contract End
CPCI Summary	4 Times	--	30 CD After PDR	30 CD After CDR 30 CD After FQT At Contract End
CPC Summary	3 Times	--	30 CD After CDR	30 CD After FQT At Contract End
Database Summary	4 Times	--	30 CD After PDR	30 CD after CDR 30 CD after FQT At Contract End
Resource Expenditure Summary	Quarterly	30 CD After End of Quarter	120 DAC	--

* DAC = Calendar Days After Contract Award
CD = Calendar Days

Figure 1-4. Model CDRL Entry

Form 1423 instruction sheet. However, a cursory explanation of DD Form 1423 in relation to the SARE requirements is provided below:

- o Block 1 contains the CDRL entry sequence number. "A005" is used in Figure 1-4 for illustration. The actual sequence number for the SARE DID in the CDRL would appear in Block 1.
- o Blocks 2 and 3 provide the title and subtitle of the SARE DID.
- o Block 4 will contain the official SARE DID number, once assigned.
- o Block 5 requires a reference to the SOW paragraph that states the contractor will provide SARE data. "SOW para. 3.4" is used for illustration.
- o Block 6 indicates the technical office that is officially requesting the data. "ESD/AC" is used for illustration.
- o "XX" appears in Block 7 to indicate that a DD Form 250, "Material Inspection and Receiving Report," is not required. "N" appears in Block 8 to indicate that CDRL requirements for approval and distribution are not applicable. Block 9 is left blank. (Block 9 is normally used when an integrating contractor is to receive a copy of the data).
- o Blocks 10 to 13 establish delivery dates. Because the SARE DID contains five forms with differing delivery requirements, the delivery dates are deferred to Block 16.
- o Block 14 lists the offices receiving copies of the data. Block 15 indicates the total number of copies to be delivered.
- o Block 16 specifies any special delivery requirements and tailoring instructions. In Figure 1-4, Block 16 refers the contractor to a CDLR back-up sheet to establish delivery dates for the five SARE forms.

The delivery dates in the model CDRL entry are geared to major program milestones when new information is naturally available to the contractor. This results in least-cost reporting while providing a time series of data to explain changes that occurred during development and impacted initial estimates.

The reporting schedule in Figure 1-4 is recommended for most programs. However, changes can be made to meet program-peculiar needs. For example, if there will be long lapses between milestones, the reporting dates can be made annual or semi-annual, rather than related to program milestones.

EVOLUTION OF THE SARE METHODOLOGY

The SARE methodology has taken on several forms since its inception. The basic requirements for a SARE data collection system were originally analyzed in 1978. The result of the analysis was a recommendation for a fully automated data collection and management system. The first draft of a proposed SARE DID was released as a working paper in December 1978 and later revised and re-released as a technical report in September 1979. As a self-contained document, the original SARE DID was comprehensive and voluminous. A decision was made in 1980 to partition the DID into two draft MIL-STDs and a simplified DID while proceeding with the development of a prototype database management system.

The first MIL-STD was the predecessor of Draft MIL-STD-X presented in this report. The second MIL-STD was a reduced version of the cost driver attributes defined in the original SARE DID. The third document, a draft DID, specified magnetic tape formats for reporting data to the government.[1]

The SARE data collection system, configured as two draft MIL-STDs, a DID and a prototype data management system, was pilot tested on an ESD program in 1982. Draft MIL-STD-X and the Draft SARE DID presented in this report evolved from their predecessors based on experiences gained during negotiations with the contractor and additional research to bring the methodology up to date with more recent cost models. The rationale behind the present configuration of the SARE documents is discussed in Sections 2 and 3.

---

[1] The original SARE DID and subsequent versions were internal reports, not in the public domain, and cannot be referenced directly by this report.

SECTION 2

OVERVIEW OF DRAFT MIL-STD-X: SOFTWARE WORK BREAKDOWN STRUCTURES FOR DEFENSE SYSTEM ACQUISITIONS

WHY A DRAFT MIL-STD?

A MIL-STD format was chosen for the SARE CWBS elements for several reasons. For one, the objective of the SARE CWBS elements is to enforce uniform cost reporting across defense system acquisition programs. A MIL-STD format is most appropriate for meeting that objective. Also, it is likely that if MIL-STD-X is adopted, it will be incorporated into a revision of MIL-STD-881A. Emulation of the structure of MIL-STD-881A will make the connection between Draft MIL-STD-X and MIL-STD-881A clearer to government personnel and contractors during initial applications.

DERIVATION OF DRAFT MIL-STD-X REQUIREMENTS

As stated in the evolution of the SARE methodology, the MIL-STD-X CWBS elements were part of the original SARE DID. They were developed by J. B. Glore of The MITRE Corporation based on his experiences supporting various ESD programs and his earlier work on the ESD Software Acquisition Management Guidebooks (Glore and Bjerstedt 1977; Glore 1977). The Draft MIL-STD-X (Attachment 1) contains within its appendices a revised version of the original SARE CWBS elements.

The "front-end" requirements of Draft MIL-STD-X were added following the test application on an ESD acquisition program to address procedural issues not covered by the original DID. These requirements were derived based on a review of the following ESD-managed acquisition programs:

- JTIDS System Exercisor
- MILSTAR
- OASIS
- TRI-TAC CNCE
- TRI-TAC CSCE
- OTH-B
- SPADOC 4
- GWEN

Several other documents addressing software WBSs were also reviewed, including (Boehm 1981), (Reiffer 1982), and two documents related to AFSC Space Division programs, (SSCAG 1980) and (Long and Toutant 1981).

KEY WBS ISSUES ADDRESSED

Several key issues had to be addressed within Draft MIL-STD-X. The following paragraphs discuss these issues and the rationale behind the decisions made by the developers. Others reviewing this report with different backgrounds and experiences may disagree with some of the decisions made. We welcome differing points of view and in Section 4 invite readers to comment using the questionnaire provided as Attachment 3.

Issue 1: Structure of Software Components in the CWBS

MIL-STD-881A provides insufficient guidance for identifying software components in the CWBS. In fact, the only mention of software is a "Computer Programs" element at level 3 under "Prime Mission Equipment" in Appendix B of MIL-STD-881A. Such a structure does not afford adequate control over software development on defense programs for which software is increasingly becoming the major cost contributor and the primary area of technical risk.

In the absence of specific guidance from MIL-STD-881A, ESD program offices and contractors have taken it upon themselves to define software WBSs. This has normally involved many iterations and has met with varying degrees of success. The WBSs observed on ESD programs appear to map into three basic structures: (1) prime mission software as a separate level 2 CWBS element with lower level breakdowns into software components, (2) software subsystems identified at level 3 under Prime Mission Equipment in parallel with corresponding hardware subsystems, and (3) as a single level 3 element under Prime Mission Equipment with a lower level breakdown into CPCIs.

It became obvious that it would be inappropriate to attempt to legislate one structure to fit the needs of all programs. Therefore, MIL-STD-X acknowledges the three generic structures discussed above and provides guidelines for selecting the one, or combination, which best meets the needs of the program.

Issue 2: Level of Software CWBS Elements

Most ESD programs reviewed partition the software system at successive levels into: (a) software subsystems, (b) computer program configuration items (CPCIs), and (c) computer program components (CPCs). On some defense programs, the CPCIs were extremely large, encompassing 50K lines of code or more. Observed CPCs ranged in size from approximately 500 to 10K lines of code. Because of this, it was decided to extend the MIL-STD-X requirements down to and including the CPC level. It is recognized that this may be appropriate for some programs but too fine grained for others. Therefore, to meet the needs of different programs, different levels of application of MIL-STD-X are expected. The final determination should be made by the procuring agency, based on the size, structure, and complexity of the particular software system.

<u>Issue 3: CPCI CWBS Elements</u>

CPCIs are the primary software products delivered on a defense program. They form the basis for functional and performance allocations, interface control, detailed specification and design, development, testing and configuration management. Clearly the CPCIs should be identified in the CWBS. However, it is not so clear what should constitute a lower level breakdown of a CPCI within the CWBS. After careful consideration, the following level 5 and 6 breakdown was determined to be appropriate to meet the visibility requirements of program managers and the data needs of software cost estimators.

Level 4	Level 5	Level 6
CPCI_n		
	Requirements Definition (Development Specification)	
	Design (Product Specification)	
	$\text{CPC}_{n.1}$	
		$\text{CPC}_{n.1}$ Design
		$\text{CPC}_{n.1}$ Code & Debug
		$\text{CPC}_{n.1}$ Integration & Checkout
	.	
	.	
	.	
	$\text{CPC}_{n.i}$	
		$\text{CPC}_{n.i}$ Design
		$\text{CPC}_{n.i}$ Code & Debug
		$\text{CPC}_{n.i}$ Integration & Checkout
	Integration & Informal Test	
	Data Generation/Conversion	
	Qualification Tests	

Level 4	Level 5	Level 6
	Documentation	
	System Test Support & Initial Maintenance	

This structure provides a comprehensive, product-oriented breakdown of a CPCI, which also corresponds to the phasing of the CPCI development. It provides consistent data collection across acquisition programs by ensuring that all activities related to the CPCI are accounted for under the CPCI. It also provides a basis for measuring the status of the CPCI development during the system acquisition, as well as a basis for future estimation of CPCI development schedule.

Issue 4: System-Level Software Activity

CPCI development accounts for only a fraction of the software-related activity on a defense program. For example, software is also a major consideration of such system level activities as System Engineering and System Test & Evaluation. The original SARE DID and the prior version of MIL-STD-X attempted to separate software-related from hardware-related activities at the system level. This was found to be artificial in practice and has been excluded from the current version of Draft MIL-STD-X.

SUMMARY OF DRAFT MIL-STD-X REQUIREMENTS

The following is a synopsis of the requirements of Draft MIL-STD-X. Each section summarizes the requirements of the corresponding Draft MIL-STD-X paragraph and, in some cases, briefly discusses why the requirement is as it is.

Scope of Application (Paragraph 1)

Two criteria are used to determine whether Draft MIL-STD-X should be applied to a defense system acquisition program:

a. MIL-STD-881A should be applied to the program, and

b. There should be a "program-unique requirement" for identification of software products in the CWBS.

The former is necessary because there would be little sense in applying MIL-STD-X to a program that does not require a CWBS in general. The latter appeals to an intentional loophole in MIL-STD-881A that allows modification of its requirements to meet program-unique requirements. Given the present state of MIL-STD-881A, nearly any defense program that includes software acquisition has a program-unique need to deviate from its exact requirements. MIL-STD-X provides a logical organization for such deviation.

Two criteria are used to determine if a program-unique requirement exists: (a) if the software contributes more than five percent to the total development cost of the system, or (b) software represents an area of major technical risk to the program. The former mandates use of MIL-STD-X on programs with significant software development effort. The latter allows its use on programs where the software effort is small in proportion to the total system cost but still represents critical technical risk.

Software Definitions (Paragraph 3)

There is currently no single authority for definitions of software terminology used in the various MIL-STDs that apply to software development on defense systems. An effort has been made to establish a set of definitions that can be applied to all types of software development. The definitions presented in Paragraph 3 of Draft MIL-STD-X have been scrutinized by a limited number of government and industry representatives and appear to be appropriate.

General Requirements (Paragraph 4)

MIL-STD-881A establishes the requirements for WBSs on defense programs. Draft MIL-STD-X builds on MIL-STD-881A so as not to duplicate its requirements or definitions. Paragraph 4 establishes this relationship.

Detailed Requirements (Paragraph 5)

The detailed requirements in Paragraph 5 of Draft MIL-STD-X address the extension of the CWBS. Six areas are addressed:

a. Placement of software components in the CWBS

b. Integration of lower-level components to form higher-level components

c. Extending the CWBS with CPCI elements

d. Tailoring the CPCI elements to apply to various types of CPCI acquisitions, including purchases, modifications, conversions, and subcontractor developments

e. Subcontract WBSs, and

f. Software development and maintenance facilities.

Paragraph 5 is essentially a mechanical procedure for extending the CWBS with software elements. The requirements are self-explanatory.

Draft MIL-STD-X Appendix A

Appendix A of Draft MIL-STD-X provides three model CWBSs to illustrate the concepts of MIL-STD-X. Each model is a generic CWBS, developed in accordance with MIL-STD-881A, which corresponds to one of the alternative methods of identifying software components discussed in MIL-STD-X Paragraph 5.1.

Draft MIL-STD-X Appendix B

Appendix B of Draft MIL-STD-X establishes and defines level 5 and 6 CWBS elements that break down the CPCIs. The contractor is instructed in MIL-STD-X Paragraph 5.3 to extend the CPCIs in the CWBS with the elements in Appendix B. The intent of Appendix B is to provide well-defined, well-structured cost reporting to support future estimation. The CWBS elements defined in Appendix B were determined to best meet these objectives.

SECTION 3

OVERVIEW OF DRAFT SARE DATA ITEM DESCRIPTION

WHY A DID?

All data to be reported by a contractor must be specified in a data item description (DID) referenced in the contract data requirements list (CDRL). Alternatives to creating a new DID would be to gather the data from government personnel monitoring acquisitions and/or attempt to glean the information from other deliverable data items.

Both alternatives would make the information secondhand, increasing the potential for error. Government monitors would waste considerable time reading superfluous material in order to collect the pertinent information. It would also be nearly impossible to maintain consistent definitions. In the end the monitor would likely confer with the contractor anyway to ensure the information is correct.

The most cost-effective way to collect quality data is to specify precisely the needed information in a DID and collect it directly from the software developers.

ORIGIN OF THE SARE DID REQUIREMENTS

The Draft SARE DID presented in Attachment 2 has been derived from the original SARE DID, a later revision of a subset of the original DID, experiences during negotiations with the pilot test contractor, and a recent review of popular software cost models and data collection forms used on other software data collection efforts.

Software cost models reviewed during various stages of SARE development include:

- Aron
- Boeing Computer Services
- Doty
- Walston & Felix
- COCOMO
- SLIM
- PRICE S
- Wolverton

The data collection forms reviewed include the NASA Software Engineering Laboratory (SEL) Data Collection Forms (DACS-A) and the DACS Productivity Data Collection Forms (DACS-B), both of which are distributed by the Data & Analysis Center for Software.

The NASA SEL forms, the COCOMO model (Boehm 1981), and the Doty model (Herd et al. 1977) were particularly useful and were used extensively to define data items in the Draft SARE DID.

KEY ISSUES ADDRESSED

The key issues addressed while developing the SARE DID and the rationale behind the decisions made are discussed below. As with Draft MIL-STD-X, we welcome differing points of view, and in Section 4 solicit comments using the questionnaire provided as Attachment 3.

Issue 1: What to Ask

In order to calibrate software cost models, the collected data must as a minimum include the software attributes used by the models to estimate cost and schedule. The database must also provide cost (that is, manhours and dollars) and the distribution of cost over time (that is, schedule and manloading profiles).

The Draft SARE DID contains five forms for collecting information on the project. The first three forms collect characteristics of the software system as a whole, the CPCIs, and the CPCs, respectively. The fourth form collects information about the size of the computer databases assembled by the contractor. The data items selected either correspond directly to software cost model attributes or are used to derive cost model attributes. The fifth form collects man-hour and dollar expenditures for software WBS elements defined in Draft MIL-STD-X.

Issue 2: Levels of Data Collection

The levels of data collection should correspond to the levels of software components in the software hierarchy. As mentioned above, the data collection forms contained in the Draft SARE DID collect information about the software system as a whole, each CPCI and each CPC.

Issue 3: Reporting Frequency

The database should provide a history of changes in the software attributes (for example, growth in software size, increased timing and storage criticality, number and relative size of engineering change proposals). This information is useful for explaining, after the fact, why actual costs differed from initial estimates. It also allows independent generation of "revised estimate at completion" during the program.

As discussed in the SARE Concept of Operation in Section 1, the reporting frequency of the SARE DID will be specified in the CDRL. The recommended frequency for the various SARE DID forms is presented in the model CDRL entry in Figure 1-4. The actual reporting requirements may vary by program. Information reported early in the program will be estimated and replaced in later reports with actual values as they become known.

SUMMARY OF THE SARE DID REQUIREMENTS

The Draft SARE DID presented in Attachment 2 contains five forms for collecting information about software development on defense system acquisition programs. The items are defined in the instruction sheets that accompany the forms.

The first three forms, the Project Summary, CPCI Summary, and CPC Summary, collect cost driver attributes for those three levels of the software hierarchy. The fourth form, the Database Summary, collects information about the computer databases. The fifth form, the Resource Expenditure Summary, is used to report expended man-hours, labor dollars, and total dollars for software CWBS elements established in accordance with Draft MIL-STD-X.

SECTION 4

SARE INDUSTRY/GOVERNMENT REVIEW

PLANS FOR A TWO-PHASE INDUSTRY/GOVERNENT REVIEW

The next steps in the SARE development are to evaluate the methodology during trial applications and disseminate the draft documents for widespread review by government, industry, and other interested organizations.

The industry/government review is planned in two phases. Phase I, which occurred during the Spring of 1983, was limited to a select set of government and industry organizations. The results of that review are summarized in the latter part of this section.

Phase II will be accomplished through general dissemination of this report. Comments on the proposed SARE documents are welcomed from all sectors of industry, government, and academia. A questionnaire has been included as Attachment 3 of this report to assist reviewers in providing their comments.

All readers are encouraged to participate. To do so, please return the completed questionnaire, and any additional comments, by 1 April 1984 to:

Headquarters, Electronic Systems Division
Director of Cost Analysis
Management and Information Systems Division
Hanscom Air Force Base, MA 01731
Attention: Capt. J. P. Dean, ESD/ACCE

Questions should be directed to Capt. Joseph P. Dean, ESD/ACCE, at (617) 861-5223 or AV 478-5223 or by mail at the above address.

Contributions to this effort will be greatly appreciated.

SUMMARY OF THE LIMITED PHASE I REVIEW

A preliminary industry/government review of the Draft MIL-STD-X was conducted during Spring 1983. The review was limited to a select set of industry and government representatives. The Draft SARE DID was not included in the review.

Thirty-five copies of Draft MIL-STD-X were distributed, and comments were received from thirteen individuals and organizations. The response was generally favorable. All specific comments dealing with phraseology and detailed requirements were considered, and those accepted have been incorporated into the current version of Draft MIL-STD-X.

General comments relating to the overall implementation of MIL-STD-X are summarized below.

Incorporation of MIL-STD-X into MIL-STD-881A

Reviewers were given a choice of recommending that Draft MIL-STD-X become a stand-alone MIL-STD, be incorporated into a revision of MIL-STD-881A, be used as guidance only, or not be pursued at all. The consensus was that Draft MIL-STD-X should be incorporated into a revision of MIL-STD-881A. Reviewers almost unanimously agreed that Draft MIL-STD-X is indeed needed and that consolidation with MIL-STD-881A would clarify the relationship and order of precedence.

MIL-STD-X was drafted as a stand-alone MIL-STD for two reasons: (1) to allow trial applications on ESD programs while it is being considered for widespread use, and (2) to simplify its incorporation into MIL-STD-881A. The ultimate form and content will be decided by the appropriate authority at some future date.

Definitions of Software Components

There was some confusion regarding the definitions of software components in Paragraph 3 of Draft MIL-STD-X. The current definitions were developed in response to the questions and suggestions of reviewers.

On defense programs, three types of software have to be addressed:

1. Prime Mission Software - all software that executes in the target computers during any mode of system operation.

2. Support Software - software that does not execute during system operation but which supports off-line test and maintenance of the defense system.

3. Other Software - software that is not a part of the defense system but is developed or acquired for such ancillary functions such as training, system engineering, and development testing.

In trying to identify these types of software in the CWBS, confusion arises over such items as operating systems, database management systems, and diagnostics programs, which can fall under either Prime Mission Software or Support Software depending on whether or not the software operates in the prime mission computers during system operation. An attempt has been made to clarify the definitions in the current version of Draft MIL-STD-X.

MIL-STD-SDS

Reviewers recommended that the Joint Logistics Commanders' new MIL-STD-SDS, "Proposed Military Standard on Defense System Software Development," associated DIDs, and proposed revisions to existing MIL-STDs be reviewed to ensure consistency. MIL-STD-SDS has been reviewed and discussions held with the contractor coordinating its revision.

It is the author's opinion that only minor changes to MIL-STD-X will be needed if MIL-STD-SDS is approved. The changes primarily involve terminology. Since MIL-STD-SDS is still in the review process and has not yet converged to its final form, the decision was made to make Draft MIL-STD-X consistent with the current MIL-STD process of software development. Adjustments can be made if and when MIL-STD-SDS is approved.

SECTION 5

CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

The major conclusions of this study are: (1) standard software data collection on defense programs is needed to eliminate the uncertainty in software cost estimation that is attributable to poor data, and (2) standard software data collection on defense programs must be fully integrated with the contractors' cost/schedule control systems.

A necessary corollary to this position is that discipline is needed in establishing and maintaining software WBSs on defense programs. This discipline is currently lacking in MIL-STD-881A.

The cost of implementing SARE data collection should not be prohibitive. Software data collection comparable to that which will be implemented by the SARE DID is now commonplace among major defense contractors. It is generally recognized that a quality database of this type is invaluable for preparing cost estimates, measuring performance, and evaluating the impact of new technologies and methodologies on software productivity.

Furthermore, because of its absence in MIL-STD-881A, software has either been ignored completely or poorly represented in CWBSs. The result has been needless iteration of CWBSs and wasted effort and expense. Draft MIL-STD-X simply adds structure to what has been a haphazard practice in the past.

The SARE methodology proposes contractual documents that can ultimately make large amounts of quality data available to the government and industry. The benefits can be increased understanding of the software development process, and increased realism in defense budgets, resulting in increased realism in contract negotiations and contract awards. Only the DoD, with the cooperation of industry, is in the position to effect such large-scale data collection.

The SARE methodology is not a panacea for software cost estimation deficiencies. However, it can provide the basis for discussion between government and industry needed to arrive at a mutually beneficial approach to software data collection.

RECOMMENDATIONS

The SARE methodology requires testing and exposure. Draft MIL-STD-X and the SARE DID should be applied to ESD programs on a trial basis beginning in FY84. The documents should also be subjected to widespread industry/government review during FY84 through public dissemination of this report.

ESD/ACC should consolidate comments from reviewers, together with initial experiences on trial applications, and report the results to Headquarters USAF and the Joint Services. ESD/ACC should include in the report a cost/benefit analysis of SARE data collection and make a recommendation regarding DoD-wide implementation.

A recommended timetable for this activity is provided below:

Oct. 1983	Begin test applications on ESD programs
April 1984	Comments due from reviewers
Sept. 1984	ESD/ACC recommendation to Hq USAF and the Joint Services regarding wide-spread implementation of SARE data collection

LIST OF REFERENCES

(Aron 1969). J. D. Aron, "Estimating Resources for Large Programming Systems," Software Engineering Techniques, NATO, Brussels, Belgium, 1969.

(Black et al. 1977). R. K. D. Black, R. P. Curnow, R. Katz, and M. D. Gray, "BCS Software Production Data," RADC-TR-77-116, Boeing Computer Services, Inc., March 1977, DTIC No. AD A039852.

(Boehm 1981). B. W. Boehm, Software Engineering Economics, Englewood Cliffs, N. J.: Prentice Hall, Inc., 1981.

(Bourdon and Duquette 1978). G. A. Bourdon and J. A. Duquette, "A Computerized Model for Estimating Software Life Cycle Costs (Model Concept)," ESD-TR-77-253, Vol. I., Hanscom Air Force Base, Mass.: AFSC Electronic System Division, April 1978, DTIC No. AD A053937.

(DACS-A). "NASA/SEL Data Collection Forms," Griffiss Air Force Base, N. Y.: Illinois Institute of Technology (IIT) Research Institute for the Data and Analysis Center for Software, Rome Air Development Center, undated.

(DACS-B). "DACS Productivity Data Collection Forms," Griffiss Air Force Base, N. Y.: Illinois Institute of Technology (IIT) Research Institute for the Data and Analysis and Center for Software, Rome Air Development Center, undated.

(DACS 1982). "The DACS Data Compendium," Griffiss Air Force Base, N. Y.: Illinois Institute of Technology (IIT) Research Institute for the Data and Analysis Center for Software, Rome Air Development Center, December 1982.

(Fleishman 1966). T. Fleishman, "Current Results from the Analysis of Cost Data for Computer Programming," System Development Corporation, August 1966.

(Freiman and Park 1979). F. R. Freiman and R. E. Park, "PRICE Software Model - Version 3: An Overview," Proceedings, IEEE-PINY Workshop on Quantitative Software Models, IEEE Catalog No. TH0067-9, October 1979.

(Glore 1977). J. B. Glore, "Software Acquisition Management Guidebook: Life Cycle Events," ESD-TR-77-22, Contract F19628-77-C-0001, Bedford, Mass.: The MITRE Corp., March 1977, DTIC No. AD A037115.

(Glore and Bjerstedt 1977). J. B. Glore and W. P. Bjerstedt, "Software Acquisition Management Guidebook: Statement of Work Preparation," ESD-TR-77-16, Contract F19628-77-C-0001, Bedford, Mass.: The MITRE Corp., January 1977, DTIC No. AD A035924.

(Herd et al. 1977). J. R. Herd, J. N. Postak, W. E. Russel, and K. R. Stewart, "Cost Estimation Study - Study Result.," RADC-TR-77-220, Vol. 1, Rockville, MD: Doty Associates, Inc., June 1977, DTIC No. AD A042264.

(Long and Toutant 1981). L. G. Long and R. P. Toutant, "Air Force Satellite Control Facility Cost Manual," Aerospace Report No. TCR-0082(2420-04)-1, Los Angeles, Calif.: The Aerospace Corp., October 1981.

(Nelson 1966). E. D. Nelson, "Management Handbook for the Estimation of Computer Programming Costs, "AD-A648750, Systems Development Corp., October 1966.

(Putnam 1978). L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Transactions on Software Engineering, July 1978.

(Reiffer 1982). D. J. Reiffer, "What Software People Do: A Work Breakdown Structure," RCI-TR-013, Torrance, Calif.: Reiffer Consultants, Inc., February 1982.

(SSCAG 1980). "Standard Work Breakdown Structure for Space Systems," unpublished report, Space Systems Cost Analysis Group, May 1980.

(Tecolote 1974). "A Provisional Model for Estimating Computer Program Development Costs," TM-7/Rev. 1, Tecolote Research, Inc., December 1974.

(Walston and Felix 1977). C. E. Walston and C. P. Felix, "A Method of Programming Measurement and Estimation," IBM Systems Journal, 16, 1, 1977.

(Wolverton 1974). R. W. Wolverton, "The Cost of Developing Large-Scale Software," IEEE Transactions on Computers, June 1984.

GOVERNMENT DOCUMENTS*

AFLCP/AFSCP 173-5, Cost/Schedule Control Systems Criteria Joint Implementation Guide

AFLCP/AFSCP 173-6, Cost Schedule Control Systems Criteria Joint Surveillance Guide

DI-F-6000C, Cost Performance Report

DI-F-6010, Cost/Schedule Status Report

DoDD 5000.1, Major System Acquisitions

DoDI 7000.1, Resource Management Systems for the Department of Defense

DoDI 7000.2, Performance Measurement for Selected Acquisitions

DoDI 7000.10, Contract Cost Performance, Funds Status and Cost/Schedule Status Reports

DoDI 7000.11, Contractor Cost Data Reporting

MIL-STD-483, Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs

MIL-STD-490, Specification Practices

MIL-STD-881A, Work Breakdown Structures for Defense Materiel Items

MIL-STD-1679(NAVY), Weapon System Software Development

MIL-STD-1521A, Technical Reviews and Audits for Systems, Equipment, Munitions and Computer Programs

MIL-STD-SDS, Joint Logistics Commanders, "Proposed Military Standard on Defense System Software Development," unofficial working paper, 15 April 1982.

*Copies of government documents are available from the Naval Publications and Forms Center, 5801 Tabor Avenue, Philadelphia, PA 19120).

GLOSSARY

AFSC	Air Force Systems Command
CDRL	Contract Data Requirements List
COCOMO	Constructive Cost Model
CPC	Computer Program Component
CPCI	Computer Program Configuration Item
C/SCSC	Cost/Schedule Control Systems Criteria
CWBS	Contract Work Breakdown Structure
DACS	Data and Analysis Center for Software
DID	Data Item Description
DoD	Department of Defense
ESD	Electronic Systems Division
JLC	Joint Logistics Commanders
MIL-STD	Military Standard
NASA SEL	NASA Software Engineering Laboratory
RFP	Request for Proposal
SARE	Software Acquisition Resource Expenditure
SDC	Systems Development Corporation
SOW	Statement of Work
WBS	Work Breakdown Structure

ATTACHMENT 1

DRAFT MILITARY STANDARD

SOFTWARE WORK BREAKDOWN STRUCTURES FOR
DEFENSE SYSTEM ACQUISITIONS

DEPARTMENT OF DEFENSE

WASHINGTON D. C. 20301

SOFTWARE WORK BREAKDOWN STRUCTURES FOR DEFENSE SYSTEM ACQUISITIONS

MIL-STD-X

1. Mandatory application. The work breakdown structure requirements established by this standard apply to all defense system acquisition programs (a) to which MIL-STD-881A is applied, and (b) for which there exists a program-unique requirement for identification of software products in the work breakdown structure. Such a program requirement exists if:

a. Five percent or more of the estimated research, development, test and evaluation (RDT&E) cost of the system is for software acquisition, and/or

b. The DoD component has determined that there is significant technical risk in the program associated with the software development.

2. Optional application. This military standard is optional for use on smaller programs to which MIL-STD-881A is not formally applied, but which are considered software intensive and/or in which there is major technical risk associated with the software.

3. Recommended corrections, additions, or deletions should be addressed to:

Headquarters, Electronic Systems Division
Director of Cost Analysis
Management and Information Systems Division
Hanscom Air Force Base, MA 01731

FOREWORD

1. The work breakdown structure (WBS) elements defined in this standard extend the requirements of MIL-STD-881A in the area of software. The standard is to be applied in conjunction with MIL-STD-881A to defense system acquisition programs on which a significant portion of the estimated development cost is for software acquisition. The principles of this standard may also be applied on smaller acquisition programs to which MIL-STD-881A is not formally applied but which are considered software critical.

2. The software WBS elements defined in this standard provide a uniform framework for:

a. Planning, budgeting, and allocating responsibilities within government organizations responsible for the acquisition of defense systems and contractor organizations responsible for their development.

b. Uniform reporting of progress and status on software efforts throughout defense system acquisition programs.

c. Consistent accumulation of resource expenditure data across defense programs that can be used to calibrate and validate software cost estimation models and methods.

3. Implementation of this standard will benefit government and industry program managers by providing a uniform structure for comparing proposals, measuring performance, and facilitating problem detection and analysis for the software portion of a defense system acquisition. It will also benefit the government and industry in general by creating a uniform software data base which will result in improved software cost estimates. This, in turn, will result in more realistic program budgeting, proposal evaluations, and contract negotiations.

TABLE OF CONTENTS

TABLE OF CONTENTS (Concluded)

LIST OF ILLUSTRATIONS

DRAFT MILITARY STANDARD

SOFTWARE WORK BREAKDOWN STRUCTURES FOR DEFENSE SYSTEM ACQUISITIONS

1. SCOPE

1.1 Purpose. This standard establishes criteria governing the preparation of software work breakdown structure elements for use in conjunction with MIL-STD-881A during the acquisition of selected defense materiel items.

1.2 Application.

1.2.1 Mandatory application. The work breakdown structure requirements established by this standard apply to all defense system acquisition programs (a) to which MIL-STD-881A is applied, and (b) for which there exists a program-unique requirement for identification of software products in the work breakdown structure. Such a program requirement exists if:

a. Five percent or more of the estimated research, development, test and evaluation (RDT&E) cost of the system is for software acquisition, and/or

b. The DoD component has determined that there is significant technical risk in the program associated with the software development.

1.2.2 Optional application. This military standard is optional for use on smaller programs to which MIL-STD-881A is not formally applied, but which are considered software intensive and/or in which there is major technical risk associated with the software.

2. REFERENCED DOCUMENTS

2.1 The following documents of the issue in effect on the date of invitation for bids or request for proposal form a part of this standard to the extent specified herein.

MIL-STD-881A, Work Breakdown Structures for Defense Materiel Items

MIL-STD-483, Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs

DI-F-6000C, Cost Performance Report

DI-F-6010, Cost/Schedule Status Report

DI-F-XXXX, Software Acquisition Resource Expenditure (SARE) Data Collection

2.2 In cases of conflict, this standard takes precedence, unless otherwise stated herein.

3. DEFINITIONS

3.1 General. Terms shall be as defined in MIL-STD-881A, MIL-STD-483, this standard, and the appendixes of this standard.

3.2 Software components. Software component is a generic term that refers to software at any level of the software hierarchy. The software components defined in the following subparagraphs are addressed by this standard. The terms used to describe the software components that apply to a particular defense system acquisition program may differ from those defined herein. However, the principles of this standard still apply.

3.2.1 Prime mission software (software system). The aggregate of all computer programs and databases that operate as part of the defense system. This includes applications software developed specifically to provide a prime mission function of the defense system and support software, such as off-the-shelf operating systems, data base management systems, on-line diagnostics, etc., which execute in the target computer(s) during any mode of system operation. The prime mission software is also referred to as the "software system" in this standard. The prime mission software may be partitioned directly into computer program configuration items or it may be partitioned into software subsystems which are in turn partitioned into computer program configuration items.

Note: The term "system segment" is not used in this standard. A system segment, as defined in MIL-STD-483, may refer to a subdivision of a defense system which has been designated for separate procurement. Under this interpretation, the same principles of this standard shall apply to system segements as to a software system as a whole. Under a broader interpretation of MIL-STD-483, a system segment may correspond to a subsystem as defined below.

3.2.2 Software subsystem. A subdivision of the software system which operates as an integral whole and provides a major function of the system. A software subsystem is comprised of two or more computer program configuration items. A software subsystem may also be a collection of computer program configuration items that are grouped together in a common classification for program management purposes (for example, support software).

3.2.3 Computer program configuration item (CPCI). An aggregation of software, or any of its discrete portions which satisfies an end use function and has been designated by the government for configuration management. CPCIs are the major software products of a system acquisition. The term "CPCI" is formally defined in MIL-STD-483. In case of conflict, MIL-STD-483 takes precedence over this paragraph.

3.2.4 Computer program component (CPC). A functionally or logically distinct part of a CPCI distinguished for convenience in designing and specifying a complex CPCI as an assembly of subordinate elements. The term "CPC" is formally defined in MIL-STD-483. In case of conflict, MIL-STD-483 takes precedence over this paragraph.

3.3 Support software. Two types of support software are acquired on defense system acquisition programs: (a) support software which operates as part of the prime mission software (that is, operating systems, database management systems, on-line diagnostic programs, etc. which execute during system operation), and (b) support software which does not operate as part of the defense system but which is used off-line to support the development, test, and maintenance of the prime mission software (that is, operating systems, compilers, linkers, loaders, simulators, debuggers, off-line diagnostic and utility programs, etc. which are used to develop the prime mission software during the system development and are delivered to maintain the system during operation, normally as part of a software development/maintenance facility). Support software which executes during any mode of system operation is considered prime mission software.

3.4 Software not included in prime mission software. Deliverable and non-deliverable software that does not operate as part of the defense system but supports a specific activity during the system acquisition phase (for example, communication network simulator used to conduct system engineering analyses, system test support software, program support libraries) or are acquired as part of such deliverable, non-prime-mission items as training, support equipment, or maintenance facilities.

4. GENERAL REQUIREMENTS

4.1 Extension of MIL-STD-881A. MIL-STD-881A establishes criteria for the preparation and use of summary, project summary, contract, and extended contract work breakdown structures (WBS) by DoD components and contractors. This standard extends the requirements of MIL-STD-881A as follows: (a) the DoD component will identify the software components in the project summary and contract WBS down to the level at which the software components have been defined prior to the release of the invitation for bids or request for proposal, (b) the DoD component will negotiate the placement of the software components in the contract WBS with the contractor, (c) the contractor shall then extend the contract WBS during the program, as the software system is defined in greater detail, to include the contract WBS elements established in this standard.

5. DETAILED REQUIREMENTS

5.1 Placement of software components in the contract work breakdown structure. The placement of software components in the contract WBS shall correspond to the configuration of the prime mission software in the defense system.

5.1.1 Prime mission software. There are three basic ways the prime mission software may be identified in the contract WBS: (a) as a separate level 2 element with lower level breakdowns into software subsystems and CPCIs, (b) as software subsystems at level 3 under prime mission equipment in parallel with corresponding hardware subsystems, or (c) as a single level 3 element under prime mission equipment with a level 4 breakdown into CPCIs. The DoD component will select the structure, or combination of structures, that best meets the requirements of the program, according to the guidelines in the following subparagraphs. The selection will be based on systems engineering analyses and negotiated with the contractor. The overall objective is to ensure that the selection of software subsystems and CPCIs corresponds to the functional breakdown of software components in the defense system. Each contract WBS discussed below is illustrated in Appendix A in the context of a generic WBS prepared in accordance with MIL-STD-881A.

5.1.1.1 Prime mission software as a separate element at level 2. Prime mission software should be identified as a separate element at level 2 of the contract WBS with a lower level breakdown into software subsystems and CPCIs if the software system is large, centralized, and partitioned into software subsystems that correspond to functional areas but not physically separate hardware subsystems (an alternative representation for software subsystems that closely parallel hardware subsystems is provided in paragraph 5.1.1.2). Figure 1 depicts a contract WBS in which prime mission software is identified at level 2. The prime mission software is partitioned at level 3 into analysis and design, integration and test, and the software subsystems. Each subsystem is broken down at level 4 into analysis and design, integration and test, and the CPCIs that comprise the subsystem.

5.1.1.2 Software subsystems in parallel with hardware subsystems. The prime mission software should be identified as separate software subsystems at level 3 under the prime mission equipment in parallel with the corresponding hardware subsystems if the defense system consists of fully distributed, independent subsystems that share few common software functions. Figure 2 depicts such a WBS. Each software subsystem is broken down at level 4 into analysis and design, integration and test, and the CPCIs that comprise the subsystem.

5.1.1.3 Software as a single element under prime mission equipment. The prime mission software should be identified as a single element at level 3 under the prime mission equipment with a level 4 breakdown into CPCIs if the software system is small and consists of individual CPCIs that are not grouped into software subsystems. Figure 3 depicts such a WBS. The software element is broken down at level 4 into analysis and design, integration and test, and the individual CPCIs.

5.1.2 Support Software. Support software which executes in the prime mission computers during any mode of system operation shall be included in the contract WBS under prime mission software. Support software which is delivered to support software maintenance during system operation, but does not execute during system operation, shall be included under software

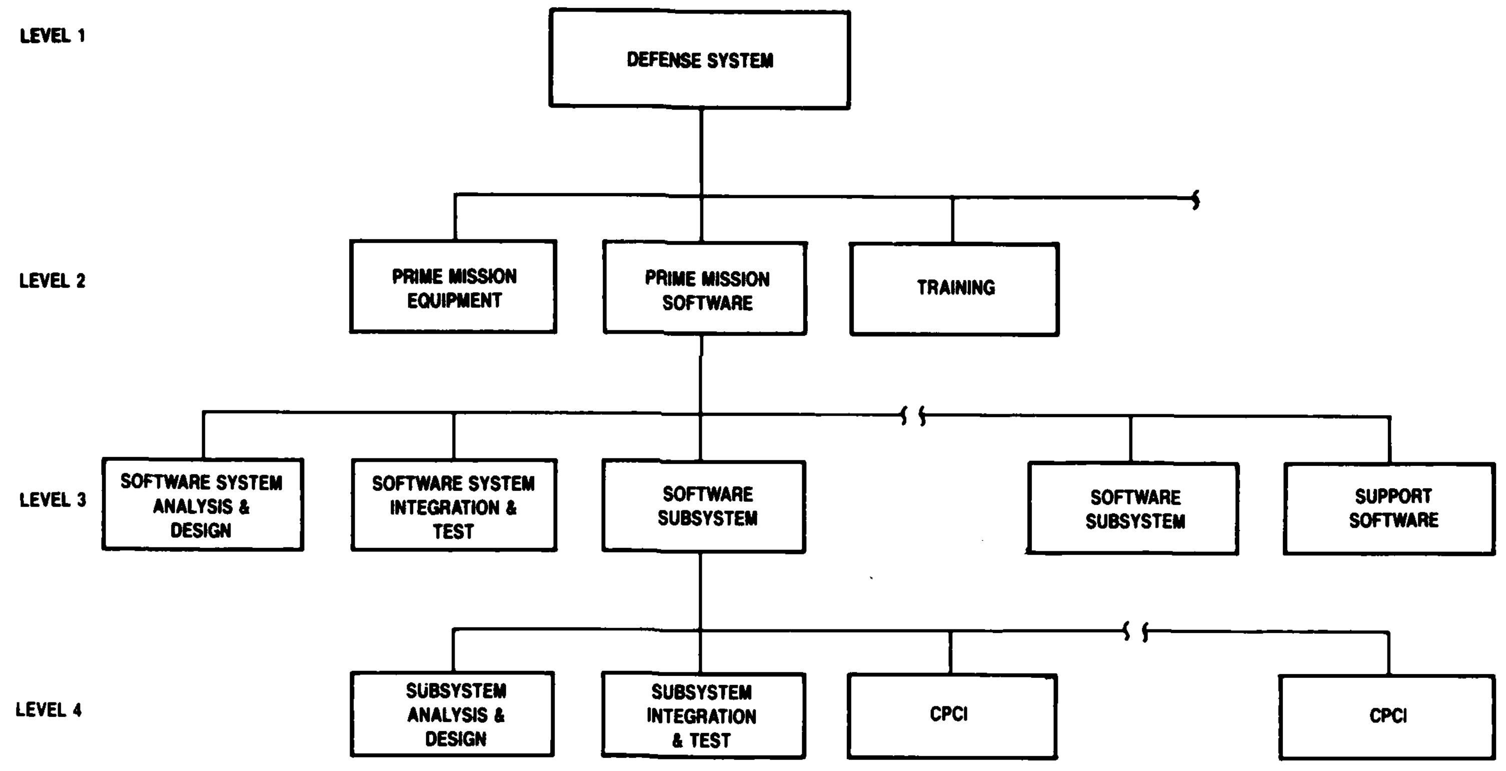


Figure 1. Prime Mission Software as a Separate Element at Level 2

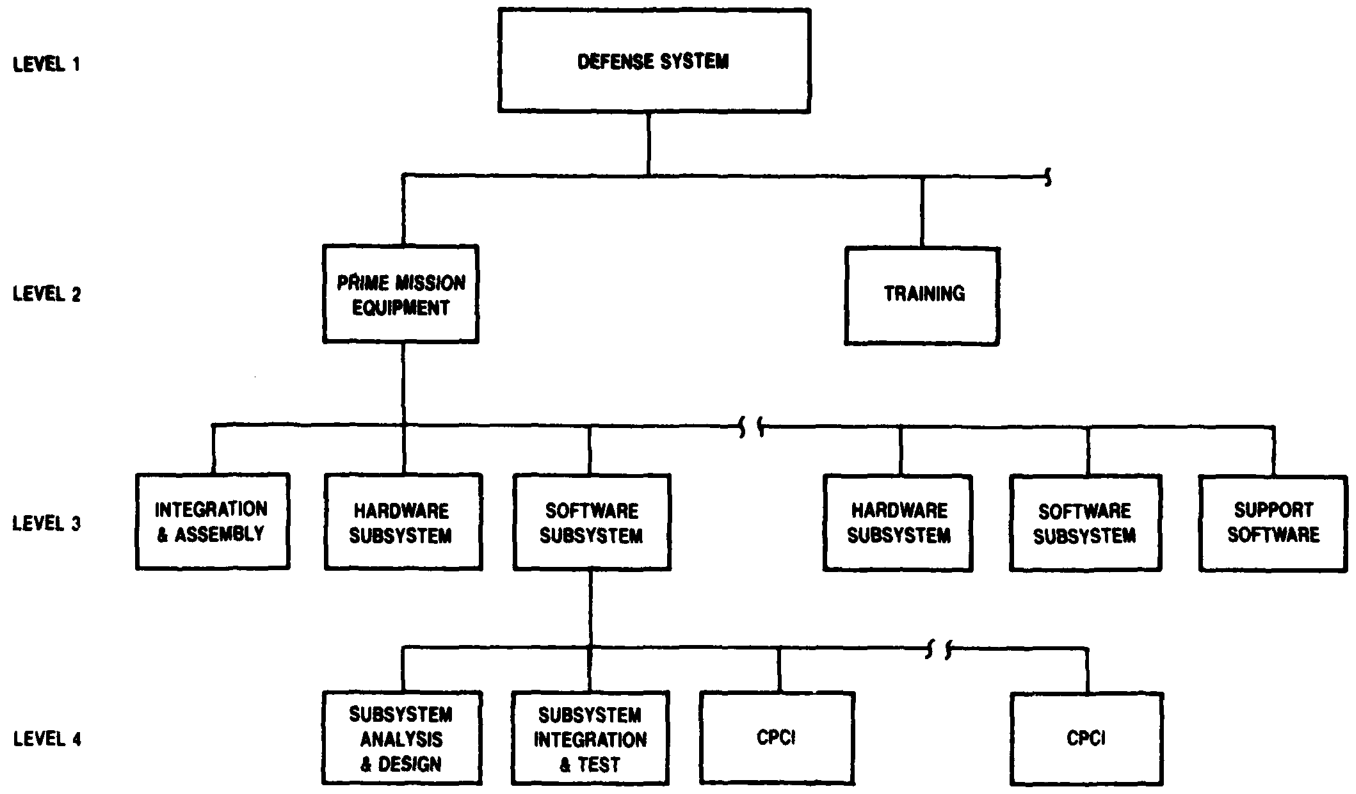


Figure 2. Software Subsystems in Parallel with Hardware Subsystems

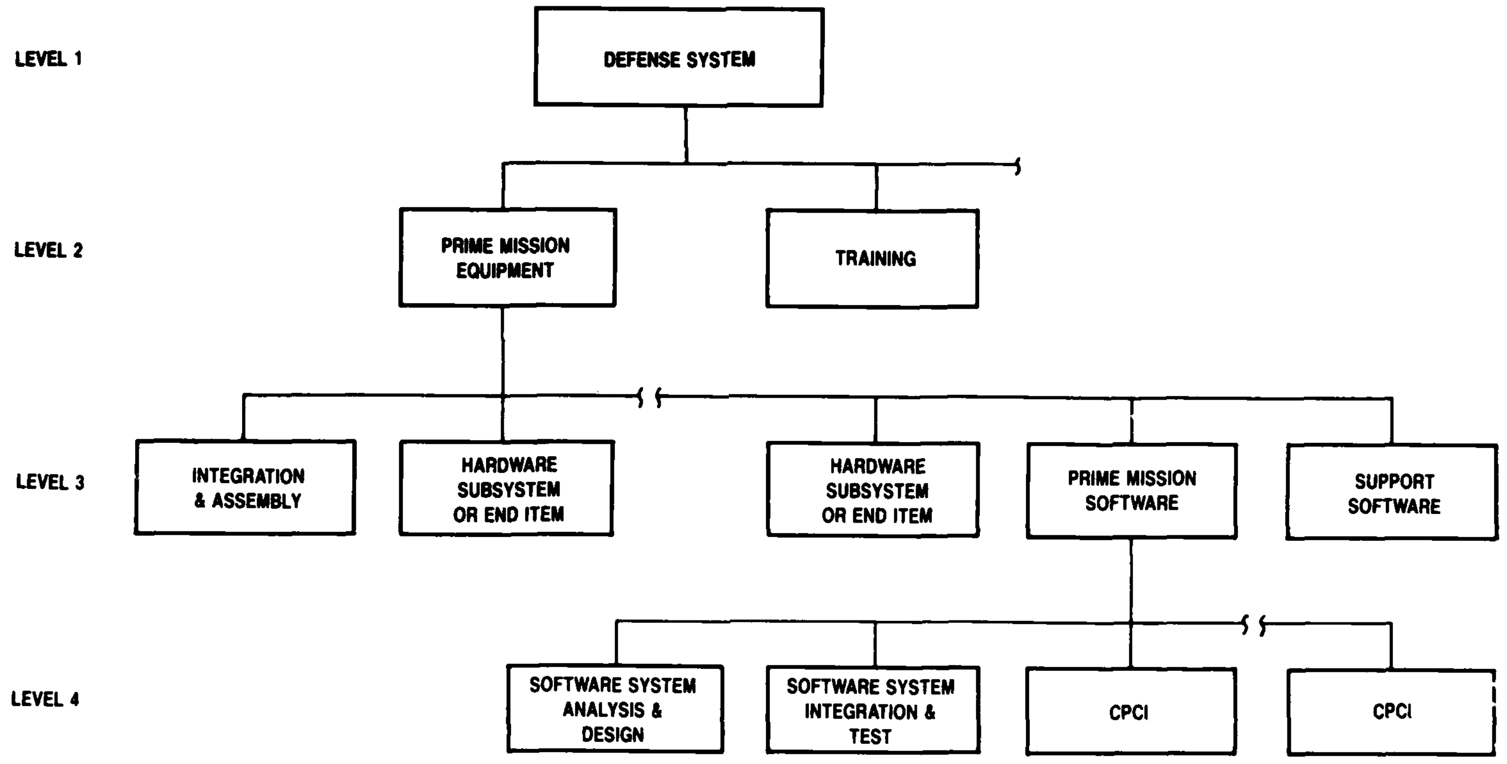


Figure 3. Software as a Single Element Under Prime Mission Equipment

development/maintenance facilities, if any. If no software development/ maintenance facilities are to be delivered with the defense system, the support software shall be included under prime mission software. Figures 1, 2, and 3 demonstrate the placement of support software under prime mission software in the contract WBS.

5.1.3 Microcode and firmware. Microcode and firmware for which the detailed design, code, and unit test are not clearly separable from the corresponding hardware item should be identified in the WBS as part of the hardware item. If the microcode/firmware is considered software in the contract and is placed under configuration management as a CPCI, the same requirements of this standard shall apply as for other CPCIs.

5.1.4 Common software. Software that is common to more than one software component should be included as part of the component that includes the software in its specification. Alternatively, if there is a significant amount of common software, it can be aggregated into a subsystem (or CPCI) that is labelled as common. This should be done only if the common software is to be specified and controlled as a separate subsystem (or CPCI).

5.1.5 Software not included in prime mission software. Software that is not a part of the prime mission software must be identified under the contract WBS element it supports. (For example, software developed solely to support system engineering analyses shall be included under System Engineering; software acquired to support off-line training shall be included under training.) This includes all such deliverable software whether or not it is placed under configuration management. This also includes all non-deliverable software which in aggregate requires 12 or more man-months to design, code, test, and document.

5.2 Software system/subsystem analysis, design, integration and test. Analysis, design, integration and test elements will be used wherever lower level software components are integrated to form a higher level software component (for example, CPCIs integrated to form a software subsystem, or software subsystems integrated to form the software system). Figures 1, 2, and 3 demonstrate the use of analysis and design and integration and test elements under the prime mission software and software subsystems.

5.2.1 Software system (subsystem) analysis and design. The technical activity undertaken to analyze and define the functional and performance requirements of the software system (subsystem), and to specify its architecture/design. This includes all studies, analyses, and specifications directly associated with the software system (subsystem) as a whole. This excludes all design and analyses directly associated with the individual lower level software components (for example, CPCIs).

5.2.2 Software system (subsystem) integration and test. The activity undertaken to integrate the software subsystems (CPCIs) into the operational software system (subsystem) and verify its correct operation prior to system level testing. This includes developing plans and procedures for integration and test, executing the procedures, analyzing the results, and preparing reports. This excludes detailed problem analysis, design, coding, and retesting associated with individual CPCIs; such activity is included under CPCI system test support and initial maintenance, defined in Appendix B. This element applies only to software activities; the integration of the software system with the prime mission equipment to form the defense system is excluded; this activity is a part of prime mission equipment integration and assembly.

5.3 Extended CPCI contract work breakdown structure elements. The contractor shall extend the contract WBS below each CPCI with the elements listed and defined in Appendix B. If the CPCI is identified at a level higher or lower than level 4, the level of the WBS elements defined in Appendix B will be adjusted accordingly. Figure 4 illustrates the breakdown of a CPCI with the elements defined in Appendix B.

5.4 Tailored use of Appendix B. The CPCI elements defined in Appendix B are required as is, except for the following.

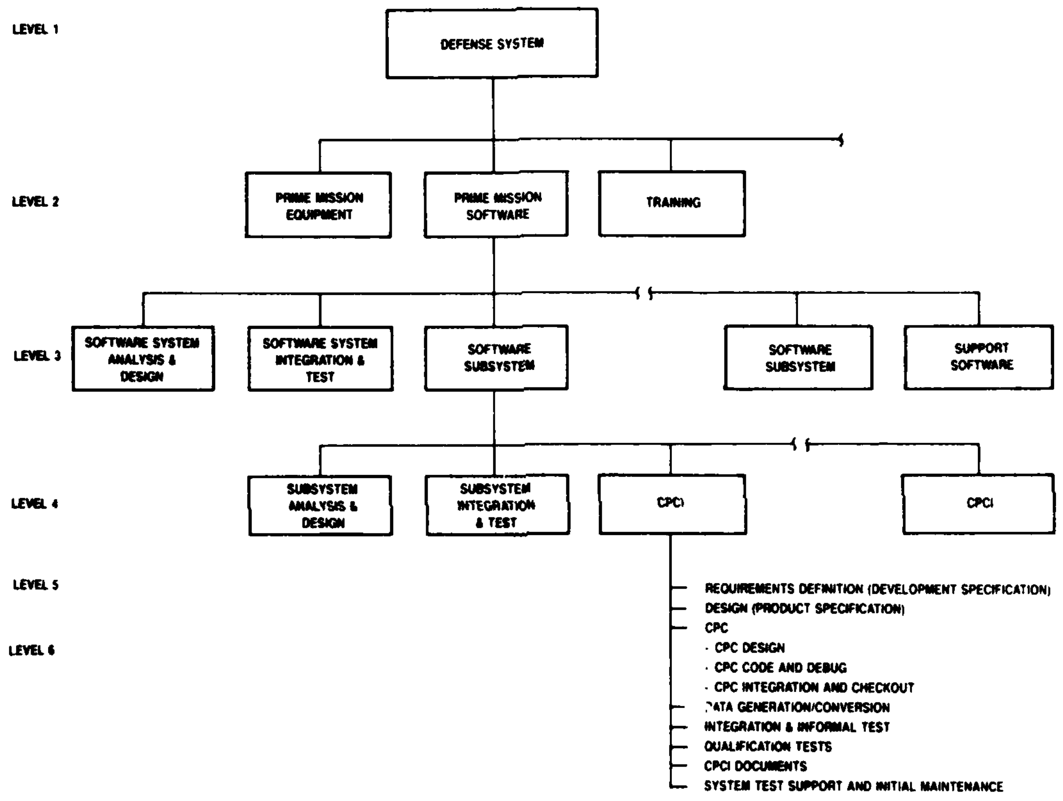
5.4.1 Addition of WBS elements. If there is activity associated with the acquisition of a CPCI that is not covered by the WBS elements in Appendix B, one or more new elements, properly defined, shall be added to account for the additional effort.

5.4.2 Exclusion of WBS elements. Those elements defined in Appendix B that do not apply to the acquisition of the CPCI shall be excluded from the extended contract WBS. If the excluded activity later becomes a part of the acquisition, the contract WBS shall be extended with the applicable element(s).

5.4.3 WBS elements for CPCIs that are not entirely new software development. The contract WBS elements in Appendix B may be tailored as follows to account for CPCIs that are not entirely new software development. The tailored application of Appendix B in these situations is subject to the approval of the DoD component.

5.4.3.1 Purchased software. Software purchased outright that requires less than 12 man-months to specify, select, install, modify, test, and document need not be extended in the WBS below the CPCI. For purchased software that requires 12 or more man-months to procure, the applicable WBS elements and definitions in Appendix B must be used. The names and definitions of the selected elements should be tailored to reflect the procurement activities. For example, it may be appropriate to break down a purchased CPCI in the contract WBS with the following elements: requirements analysis, package specification, package selection and purchase, package installation, package modification (if needed), acceptance test, and documentation.

5.4.3.2 Modification/conversion of existing software. Modification/ conversion of existing software that requires less than 12 man-months to study, specify, integrate, test, document, and modify/convert existing documentation need not be extended in the contract WBS below the CPCI. For software modification/conversion that requires 12 or more man-months to



LEVEL 1
LEVEL 2
LEVEL 3
LEVEL 4
LEVEL 5
LEVEL 6
DEFENSE SYSTEM
PRIME MISSION EQUIPMENT
PRIME MISSION SOFTWARE
TRAINING
SOFTWARE SYSTEM ANALYSIS & DESIGN
SOFTWARE SYSTEM INTEGRATION & TEST
SOFTWARE SUBSYSTEM
SOFTWARE SUBSYSTEM
SUPPORT SOFTWARE
SUBSYSTEM ANALYSIS & DESIGN
SUBSYSTEM INTEGRATION & TEST
CPCI
CPCI
REQUIREMENTS DEFINITION (DEVELOPMENT SPECIFICATION)
DESIGN (PRODUCT SPECIFICATION)
CPC
· CPC DESIGN
· CPC CODE AND DEBUG
· CPC INTEGRATION AND CHECKOUT
DATA GENERATION/CONVERSION
INTEGRATION & INFORMAL TEST
QUALIFICATION TESTS
CPCI DOCUMENTS
SYSTEM TEST SUPPORT AND INITIAL MAINTENANCE

Figure 4. CPCI Work Breakdown Structure Elements

complete, the applicable contract WBS elements and definitions from Appendix B must be used. The names and definitions of the selected elements should be tailored to reflect the modification/conversion activities. For example, it may be appropriate to break down a software modification in the contract WBS with elements identical to those in Appendix B while it may be appropriate to use the following elements to break down a software conversion: feasibility analysis, conversion plans and procedures, conversion implementation, integration and test, and documentation conversion.

5.4.3.3 Subcontractor developed software. The contract WBS elements and definitions in Appendix B that apply to the prime contractor activities related to subcontractor developed CPCIs must be included in the prime contract WBS under the respective CPCIs. For example, the prime contractor may be responsible for preparing the development specification and conducting qualification testing of each subcontractor developed CPCI. (The application of this standard to subcontract WBSs is discussed in paragraph 5.5.)

5.5 Subcontract work breakdown structures. The prime contractor shall identify in the prime contract WBS those CPCIs which are being acquired from subcontractors. Each CPCI shall be broken down into the prime contractor products, per paragraph 5.4.3.3, and a summary element for the subcontractor effort. The prime contractor may negotiate any subcontract WBS providing it identifies the CPCIs at a cost reporting level, and the lower level breakdowns of the CPCIs cover the same products and activities as the contract WBS elements defined in Appendix B of this standard.

5.6 Software development and maintenance facilities. Software development and maintenance facilities that are deliverable as part of the system acquisition will be identified at level 2 of the contract WBS with a level 3 breakdown into the following elements: equipment, services, facilities (that is, brick and mortar type construction), and initial operation and maintenance. Equipment will be broken down at level 4 into an integration and assembly element and the individual hardware and software end items that are acquired specifically for the facility. If the software end items are placed under configuration management, the same requirements of this standard shall apply as for other CPCIs.

6. MISCELLANEOUS

6.1 Financial data reporting. The purpose of MIL-STD-X is to augment the contract WBS requirements of MIL-STD-881A in the area of software development. MIL-STD-X can be used in conjunction with DI-F-6000C or DI-F-6010 to measure contractor performance during a defense system acquisition program. MIL-STD-X can also be used in conjunction with DI-F-XXXX to collect resource expenditure data for the purpose of estimating software development costs for future systems.

APPENDIX A

IDENTIFICATION OF SOFTWARE COMPONENTS IN A WORK BREAKDOWN STRUCTURE

10. SCOPE

10.1 This appendix illustrates the placement of software components in a work breakdown structure (WBS). This appendix is for illustration only (refer to paragraph 5.1).

20. PRIME MISSION SOFTWARE AS A SEPARATE ELEMENT AT LEVEL 2

20.1 The following generic WBS, prepared in accordance with MIL-STD-881A, illustrates the placement of prime mission software as a separate element at level 2 in the WBS. Prime mission software is broken down at level 3 into analysis and design, integration and test, and the software subsystems. Each software subsystem is broken down at level 4 into subsystem analysis and design, subsystem integration and test, and the CPCIs that comprise the subsystem.

20.2 The other level 2 and 3 WBS elements defined in MIL-STD-881A have been included for reference only and are not a part of this standard.

20.3 The hardware and software subsystems at level 3 and the CPCIs at level 4 will be expanded by the DoD component or the contractor into the particular subsystems and CPCIs that apply to the defense system.

Level 1	Level 2	Level 3	Level 4
Defense System			
	Prime Mission Equipment		
		Integration and Assembly	
		Hardware Subsystem or End Item	
		•	
		•	
		•	
		Hardware Subsystem or End Item	

PREVIOUS PAGE IS BLANK

Level 1	Level 2	Level 3	Level 4
	Prime Mission Software		
		Software System Analysis and Design	
		Software System Integration and Test	
		Software Subsystem $_1$	
			Subsystem Analysis and Design
			Subsystem Integration and Test
			CPCI
			.
			.
			.
			CPCI
		.	
		.	
		.	
		Software Subsystem $_n$	
			Subsystem Analysis and Design
			Subsystem Integration and Test
			CPCI
			.
			.
			.
			CPCI
		Support Software	
			CPCI
			.
			.
			.
			CPCI
	Training*		
		Equipment	
		Services	
		Facilities	

Level 1	Level 2	Level 3	Level 4
	Peculiar Support Equipment*		
		Organizational/ Intermediate	
		Depot	
	Systems Test and Evaluation*		
		Development Test and Evaluation	
		Operational Test and Evaluation	
		Mockups	
		Test and Evaluation Support	
		Test Facilities	
	System/Program Management*		
		Systems Engineering	
		Project Management	
	Data*		
		Technical Publications	
		Engineering Data	
		Management Data	
		Support Data	
		Data Depository	
	Operational/ Site Activation*		
		Contractor Technical Support	
		Site Construction	
		Site/Ship/Vehicle Conversion	
		System Assembly,	

Level 1	Level 2	Level 3	Level 4
		Installation and Checkout on Site	
	Common Support Equipment*		
		Organizational/ Intermediate	
		Depot	
	Industrial Facilities*		
		Construction/Conversion/ Expansion	
		Equipment Acquisition or Modernization	
		Maintenance	
	Initial Spares and Initial Repair Parts		

30. SOFTWARE SUBSYSTEMS IN PARALLEL WITH HARDWARE SUBSYSTEMS

30.1 The following generic WBS identifies the prime mission software as separate subsystems in parallel with the hardware subsystems.

30.2 The hardware subsystems, software subsystems and CPCIs will be expanded by the contractor into the particular subsystems, and CPCIs that apply to the defense system.

Level 1	Level 2	Level 3	Level 4
Defense System			
	Prime Mission Equipment		
		Integration and Assembly	
		Hardware Subsystem $_1$	
		Software Subsystem $_1$	
			Subsystem Analysis and Design

Level 1	Level 2	Level 3	Level 4
			Subsystem Integration and Test
			CPCI
			⋮
			CPCI
		⋮	
		Hardware Subsystem $_n$	
		Software Subsystem $_n$	
			Subsystem Analysis and Design
			Subsystem Integration and Test
			CPCI
			⋮
			CPCI
		Support Software	
			CPCI
			⋮
			CPCI
	Training*		
		Equipment	
		Services	
		Facilities	

Level 1	Level 2	Level 3	Level 4
	Peculiar Support Equipment*		
		Organizational/ Intermediate	
		Depot	
	Systems Test and Evaluation*		
		Development Test and Evaluation	
		Operational Test and Evaluation	
		Mockups	
		Test and Evaluation Support	
		Test Facilities	
	System/Program Management*		
		Systems Engineering	
		Project Management	
	Data*		
		Technical Publications	
		Engineering Data	
		Management Data	
		Support Data	
		Data Depository	
	Operational/ Site Activation*		
		Contractor Technical Support	

Level 1	Level 2	Level 3	Level 4
		Site Construction	
		Site/Ship/Vehicle Conversion	
		System Assembly, Installation and Checkout on Site	
	Common Support Equipment*		
		Organizational/ Intermediate	
		Depot	
	Industrial Facilities*		
		Construction/Conversion/ Expansion	
		Equipment Acquisition or Modernization	
		Maintenance	
	Initial Spares and Initial Repair Parts*		

40. SOFTWARE AS A SINGLE ELEMENT UNDER PRIME MISSION EQUIPMENT

40.1 The following generic WBS identifies the prime mission software as a single level 3 element under prime mission equipment with a level 4 breakdown into analysis and design, integration and test, and the CPCIs that comprise the software system.

40.2 The hardware subsystems and CPCIs will be expanded by the DoD component or the contractor into the particular subsystems and CPCIs that apply to the defense system.

<u>Level 1</u>	<u>Level 2</u>	<u>Level 3</u>	<u>Level 4</u>
Defense System			
	Prime Mission Equipment		
		Integration and Assembly	
		Hardware Subsystem or End Item	
		⋮	
		Hardware Subsystem or End Item	
		Prime Mission Software	
			Software System Analysis and Design
			Software System Integration and Test
			CPCI
			⋮
			CPCI
		Support Software	
			CPCI
			⋮
			CPCI
	Training*		
		Equipment	
		Services	
		Facilities	

Level 1	Level 2	Level 3	Level 4
	Peculiar Support Equipment*		
		Organizational/ Intermediate	
		Depot	
	Systems Test and Evaluation*		
		Development Test and Evaluation	
		Operational Test and Evaluation	
		Mockups	
		Test and Evaluation Support	
		Test Facilities	
	System/Program Management*		
		Systems Engineering	
		Project Management	
	Data*		
		Technical Publications	
		Engineering Data	
		Management Data	
		Support Data	
		Data Depository	
	Operational/ Site Activation*		
		Contractor Technical Support	
		Site Construction	

Level 1	Level 2	Level 3	Level 4
		Site/Ship/Vehicle Conversion	
		System Assembly, Installation and Checkout on Site	
	Common Support Equipment*		
		Organizational/ Intermediate	
		Depot	
	Industrial Facilities*		
		Construction/Conversion/ Expansion	
		Equipment Acquisition or Modernization	
		Maintenance	
	Initial Spares and Initial Repair Parts		

*Computer programs may be acquired or developed to support these activities. Such software must be identified in the CWBS, per paragraph 5.1.5.

APPENDIX B

WORK BREAKDOWN STRUCTURE ELEMENTS
FOR COMPUTER PROGRAM CONFIGURATION ITEM DEVELOPMENT

10. SCOPE

10.1 This appendix establishes and defines work breakdown structure (WBS) elements for computer program configuration item (CPCI) development.

20. CPCI WBS ELEMENTS

20.1 The following level 4, 5, and 6 WBS elements apply to a CPCI development. The level at which a CPCI appears in the contract WBS will be determined per paragraph 5.1. If a CPCI appears at a level higher or lower than level 4, the levels of the WBS elements defined below will be adjusted accordingly.

20.2 Tailoring of this appendix for CPCIs that are not entirely new software development is addressed in paragraph 5.4.

20.3 The computer program components (CPCs) presented at level 5 will be expanded by the contractor into the particular CPCs that apply to the CPCI.

<u>Level 4</u>	<u>Level 5</u>	<u>Level 6</u>
Computer Program Configuration Item (CPCI)		
	CPCI Requirements Definition (Development Specification)	
	CPCI Design (Product Specification)	
	Computer Program Component (CPC)	
		CPC Design
		CPC Code & Debug
		CPC Integration & Checkout
	•	
	•	
	•	

<u>Level 4</u>	<u>Level 5</u>	<u>Level 6</u>
	Computer Program Component (CPC)	
		CPC Design
		CPC Code & Debug
		CPC Integration & Checkout
	CPCI Data Generation/ Conversion	
	CPCI Integration and Informal Test	
	CPCI Qualification Tests	
	CPCI Related Documents	
	CPCI System Test Support and Initial Maintenance	

30. DEFINITIONS

30.1 <u>CPCI Requirements Definition (Development Specification)</u>. The software-related engineering and design activity undertaken to define, evaluate, and revise as necessary the CPCI requirements. This activity results in the Computer Program Development Specification for the CPCI.

30.1.1 The <u>definition</u> effort includes specifying CPCI requirements such as the following:

a. The structure and logic of the application (for example, mathematical formulation)

b. Bounds on design parameters (that is, ranges of parameter values for which the CPCI gives valid results)

c. Input/output message types, formats, and data rates

d. Database requirements (for example, data types, structures, data element characterization)

e. Performance (for example, accuracy, execution timing, and response times as functions of workloads) in normal and failure modes

f. Built-in test for failure identification

g. Transition to and from failure modes

h. Provisions for future growth (for example, extra memory, auxiliary storage, channel capacity, and processor capacity)

i. Classified data (if any)

j. Man-machine interface (if any)

k. Interfaces with other CPCIs, systems, or segments

l. Government-furnished software that the CPCI must incorporate

m. Site adaptation parameters

n. Main and auxiliary memory allocations

o. Testing requirements against which detailed CPCI test plans and procedures must be written.

30.1.2 The <u>evaluation</u> activity includes:

a. Examining new or revised CPCI requirements for clarity, consistency, completeness, testability, relative worth, and traceability to the system or segment specification

b. Reporting identified problems to those responsible for their correction.

30.1.3 The <u>requirements revision</u> activity includes:

a. Identifying and assessing the technical, schedule, and cost impacts of proposed changes to the CPCI (or other CIs, subsystems, or the system) on the CPCI requirements, design, coding, test, integration, or documentation

b. Preparation and review of formal engineering change proposals and specification change notices or their equivalents.

30.2 <u>CPCI Design (Product Specification)</u>. The activity undertaken to prescribe the structure and functions of a CPCI and the methods of its implementation; to assess this structure and methodology for correctness, completeness, simplicity, and efficiency; and to revise it as necessary. This activity normally entails drafting the CPCI Computer Program Product Specification to specify the detailed CPCI design prior to coding, revising the Product Specification during the CPCI development, and then finalizing the Product Specification after the CPCI has been tested. The results of this activity typically include:

a. CPCI decomposition:

(1) identification of each Computer Program Component (CPC)

(2) allocation of the CPCI functional requirements among the CPCs

(3) specification of each CPC's inputs, outputs, and functions

(4) detailed definition of the interfaces among the CPCs

(5) flowcharts or other graphical representation of the logic of CPC interaction.

(6) Program listings

b. Implementation methodology:

(1) allocation of storage among modules and assemblies

(2) common modules to be used

(3) techniques to be used to design the software in accordance with specified constraints and standards

c. Other:

(1) data structures (for example, files, tables, parameters)

(2) service of interrupts

(3) algorithms (for example, logic, formulas) given in the mathematical formulation and in other requirements

(4) standards for code structure

(5) parameter boundary conditions (that is, range of parameter values over which the CPCI gives valid results)

(6) error handling and reporting

(7) initial system start-up, system shutdown and recovery.

d. Assessment of the original or revised CPCI design for correctness, completeness, simplicity, and efficiency. This assessment may include simulation of selected CPCI logic to generate input and output mappings, plus execution time data, which can be compared to the CPCI functional and performance requirements.

30.3 Computer Program Component (CPC). The detailed design, implementation (coding, compilation and debugging of lower level modules), integration (assembly of lower level modules into the operational CPC), and checkout of the CPC to verify its correctness and proper performance as a unit. This activity ends when the CPC is released to be integrated with other CPCs to form the CPCI.

30.3.1 CPC Design. The activity undertaken to prescribe the method of the CPC implementation to the level of detail necessary before the start of coding; to assess this prescription for correctness, completeness, simplicity, and efficiency; and to revise the prescription as necessary. This activity encompasses:

a. Identifying the modules and subassemblies of the CPC

b. Allocating functions to each module

c. Specifying module input and output

d. Diagramming module interaction logic

e. Allocating storage among modules

f. Identifying common modules

g. Identifying CPC data structures (for example, files, tables, parameters)

h. Verifying selected CPC logic (for example, generating input/output mappings, generating execution time data and comparing it to specified execution times)

i. This element excludes all coding activity, even if such activity results in direct translation of performance requirements into source code.

30.3.2 CPC Code and Debug. The activity undertaken to implement the design of the CPC in software including:

a. Learning the latest version of the related Product Specification (or equivalent)

b. Learning the programming language to be used

c. Translating CPC module design into source language instructions and data per the design specified in the latest version of the Computer Program Product Specification (or equivalent)

d. Generating or modifying the source instructions needed to integrate the set of coded modules comprising the CPC

e. Compiling the code and correcting compilation errors

f. Reviewing code by the responsible programmers, by peers, and/or via symbolic execution of the logic

g. Correcting code defects

h. Adding comments to source listings during the generation of source instructions.

i. Assuring compliance with programming documentation standards.

30.3.3 CPC Integration and Checkout. The activity undertaken to integrate the modules that comprise the CPC into an operational whole, and plan and conduct tests that verify the correct performance of the parts of the CPC and the proper performance of the CPC operating as a unit. This activity includes integration and test planning, procedure development, test data preparation, determination of expected results, test execution, error identification, and data reduction. This element excludes the modification of the CPC to correct errors identified during integration and checkout; such activity is encompassed by CPC Code and Debug.

30.4 CPCI Data Generation/Conversion. The activity undertaken to generate, or convert into a new character representation or format, data that is read and/or manipulated by the CPCI while carrying out its function. This excludes data structures developed as part of the CPCI to control its operation.

30.4.1 Data generation activities include:

a. Analyzing the CPCI data requirements by reviewing the latest version of the CPCI Computer Program Product Specification (or equivalent)

b. Designing the physical data structures

c. Analyzing documents necessary to determine the content of the data records

d. Translating the data requirements into data records

e. Entering, formatting, and storing the data records in machine readable other form

f. Devising and running tests to ensure that the data records are correct and complete

g. Documenting the data structure and contents of the data records.

30.4.2 Data conversion encompasses the activity undertaken to convert existing data to the form or format needed by the CPCI. This activity includes:

a. Translating character codes

b. Reformatting data records

c. Sorting translated data records

d. Devising and running tests to ensure correct and complete conversion results

e. Documenting the relationship between the original and translated data.

f. This activity excludes the conversion of existing code to a new source language or a new version of an existing source language. Such activity is encompassed by CPC Code and Debug.

30.5 CPCI Integration and Informal Test. The activity undertaken to plan and conduct tests that verify correct and proper performance of the CPCI operating as a whole prior to formal qualification testing. Although government personnel are sometimes contractually allowed to observe these tests, the tests do not normally require government approval of plans, procedures and results, as do formal qualification tests.

30.5.1 The planning activity encompasses:

a. Defining test scope and objectives

b. Establishing the test approach, acceptance criteria, verification methods, order of integration, inputs, and methods to record results

c. Establishing test locations, schedules, and responsibilities of those involved.

30.5.2 The conduct and analysis activity encompasses:

a. Developing test procedures

b. Preparing test data and expected results

c. Executing the test procedures and recording test results

d. Reducing test results (for example, calculation of performance
parameters from test results), identifying errors and preparing test data sheets

e. Reporting results.

30.6 CPCI Qualification Tests. The activity undertaken to devise and revise as necessary plans and procedures for testing the CPCI against its Computer Program Development Specification (or equivalent), to exercise those procedures, collect test results, analyze the results, identify problems, and report test results. This includes all activity related to the Preliminary Qualification Tests (PQT) and Formal Qualification Tests (FQT) of the CPCI. This activity excludes investigation and isolation of problems, formulation of computer program design and code changes, and implementation of computer program changes. These activities are encompassed by the WBS elements under which these efforts were originally performed.

30.6.1 The test planning activity encompasses:

a. Defining test objective and scope

b. Establishing the test approach, acceptance criteria, verification methods, data recording methods, and data reduction methods

c. Establishing test locations, schedules, and responsibilities of those involved.

30.6.2 Test procedures include the activity undertaken to draft and subsequently update the detailed procedures for verifying the correct operation and satisfactory performance of the CPCI in relation to the Computer Program Development Specification (or equivalent) test requirements and test plans. This activity typically involves specifying:

a. Operator actions and expected responses

b. Test inputs and expected results

c. Data tables to be used

d. Data reduction methods

e. Support software such as special test data recording software, playback software, data reduction software and other software needed to supply test input, record or reduce test output, or to control test sequencing

f. Simulators of external equipment or hardware subsystems interfacing with the computer to be used to test the CPCI

g. Cross-references to the CPCI test plan and Computer Program Development Specification (or equivalent) test.

30.6.3 Test conduct and analysis includes the activity undertaken to perform iterative runs of the CPCI qualification tests (for example, PQT and FQT). This activity includes carrying out the formal test procedures in the presence of the government, collecting and reducing test results, analyzing the test results, and reducing data to verify correct performance (or detect errors). The data reduction activity encompasses:

a. Inserting into test data sheets test results obtained from playback of automatically-recorded data, produced during execution of the formal CPCI tests

b. Calculating and transforming recorded data into forms that can be compared to expected results, where such calculations and transformations may be done by hand or with computer program support.

30.6.4 The analysis and error detection activities include:

a. Identifying discrepancies between expected and observed CPCI qualification test results

b. Diagnosing each discrepancy

c. Suggesting changes to the CPCI requirements, design, implementation documentation, or test procedures necessary to correct detected errors.

30.6.5 Test report preparation includes the engineering activity undertaken to draft and revise as necessary the formal qualification test reports. The test report typically includes:

a. Completed test data sheets

b. Data reduction results

c. Test logs

d. Detected errors

e. Suggested corrections to requirements, design, implementation methods, and test procedures necessary to correct detected errors

f. Recommendations for additional testing

g. Conclusions.

Note: This element excludes the activity undertaken to transform the test report format from contractor format to the format specified in the CDRL. Such cost is encompassed by level 2 WBS element Data.

30.7 CPCI Related Documents. The engineering effort to prepare CPCI-related documents that are in addition to the development and product specifications, test plans, test procedures, test reports, and other documents which are the direct result of the other CPCI elements. This includes such activity as the preparation of users' manuals, CPCI maintenance documents, etc.

Note: This activity excludes such effort that can be reduced or will not be incurred if the corresponding data item is eliminated from the Contract Data Requirements List (CDRL). Such costs are encompassed by the level 2 WBS element called "Data".

30.8 CPCI System Test Support and Initial Maintenance. The activity undertaken to modify the CPCI design and code, informally test the modifications, retest the CPCI, and revise the CPCI specifications and other documentation to resolve problems identified during system and subsystem level integration and test. This includes corrective maintenance of the CPCI conducted by the contractor prior to responsibility transfer to the government.

ATTACHMENT 2

DATA ITEM DESCRIPTION	2. IDENTIFICATION NO(S) AGENCY	NUMBER
1. TITLE Software Acquisition Resource Expenditure (SARE) Data Collection	USAF	
3. DESCRIPTION/PURPOSE a. This data item description (DID) is one of two documents that comprise the Software Acquisition Resource Expenditure (SARE) data collection methodology. The other is MIL-STD-X, "Software Work Breakdown Structures for Defense System Acquisitions." This DID collects technical information about software cost drivers to supplement cost/schedule data reported for software-related WBS elements established by MIL-STD-X. The purpose of the SARE methodology is to (Continued on Page 2)	4. APPROVAL DATE TBD 5. OFFICE OF PRIMARY RESPONSIBILITY AFSC/ESD 6. DDC REQUIRED TBD	
7. APPLICATION/INTERRELATIONSHIP 7.1 Application This DID applies to defense system acquisition programs to which MIL-STD-881A (Work Breakdown Structures for Defense Materiel Items) and MIL-STD-X (Software Work Breakdown Structures for Defense System Acquisitions) are applied. 7.2 Interrelationship This DID provides technical information on software cost drivers to supplement cost/schedule data reported on DI-F-6000C (Cost Performance Report) or DI-F-6010 (Cost/Schedule Status Report) for software-related WBS elements established in accordance with MIL-STD-881A and MIL-STD-X.	8. APPROVAL LIMITATION TBD 9. REFERENCES (Mandatory as cited in block 10) DI-F-6000C DI-F-6010 MIL-STD-881A MIL-STD-X (Continued on Page 2) MCSL NUMBER(S)	

10. PREPARATION INSTRUCTIONS

a. The contractor shall report the information specified on the forms contained herein according to the schedule specified in the CDRL. Instructions for completing each form and the definitions of individual data items are provided on the instruction pages.

b. The direct labor hours, direct labor dollars, and total dollars reported on the Resource Expenditure Summary shall be from WBS elements established in accordance with MIL-STD-881A and MIL-STD-X. The reporting levels are specified in the CDRL.

c. Each form shall be signed and dated by the person(s) preparing the information and the person approving it for delivery. The CDRL reporting milestone for which the information is being prepared shall also be identified.

DD FORM 1 JUN 68 1664

3. Description/Purpose (Continued)

create a database which can be used to develop, calibrate, and maintain software cost estimating models for defense programs and to measure the impact of new technologies, tools, and techniques on software quality and productivity.

b. This DID contains five forms for reporting information about the software and software development environment on a defense system acquisition program:

(1) Program Summary
(2) CPCI Summary
(3) CPC Summary
(4) Database Summary
(5) Resource Expenditure Summary

These forms have been derived from the parameters of popular software cost estimating models, the COCOMO model in particular (Boehm 1981), and from data collection forms used by the NASA Software Engineering Laboratory (NASA SEL). The sources are referenced wherever data item definitions have been taken with little or no modification.

9. References (Continued)

(Boehm 1981). B. W. Boehm, Software Engineering Economics, Englewood Cliffs, N. J.: Prentice Hall, Inc., 1981

(NASA SEL). "NASA/SEL Data Collection Forms," Griffiss Air Force Base, N. Y.: Illinois Institute of Technology Research Institute, Data and Analysis Center for Software (DACS) for the Rome Air Development Center, undated

INSTRUCTIONS FOR COMPLETING THE PROJECT SUMMARY FORM

This form is used to provide information about the project as a whole that affect the software development. Information reported early in the project shall be estimated as accurately as practicable. Intermediate reports shall contain actuals if known and updated estimates. The final report shall contain all actual data. In cases where the reporting of actual data is limited by measurement accuracy, the reported value should reflect at least 90 percent confidence. Data items shall be continued on separate pages if additional space is needed.

1. PROJECT DESCRIPTION

1.1 List of Interfacing Systems. List other systems with which the system under development must communicate. Indicate if an interfacing system is concurrently under development.

1.2 Major Software Products. Describe the major deliverable software products of the project.

2. RESOURCES

Reusable Items From Similar Projects. List previous projects that will contribute to the software developed on this project. For each, indicate the approximate number of deliverable source instructions (as defined in 3.1 below) that will be adapted to the current project. Indicate the approximate percentage of the adapted software's design which must be modified to adapt to the new objectives and environment. Indicate the percent of the adapted software's code which must be modified. Also, indicate the approximate percentage of effort required to integrate and test the adapted software compared to the normal amount of integration and test effort for a new development of comparable size and difficulty. The percentages of design modification, code modification, and integration may be greater than 100 percent if the effort required is greater than the effort which would have been needed to develop the software from scratch. (Boehm 1981), (NASA SEL)

3. TOTAL SYSTEM SIZE

3.1 Deliverable Source Instructions. Indicate the total number of deliverable source instructions in the software system, excluding source lines that are entirely source code documentation and source instructions from unmodified utility software. Include job control language instructions, format statements, and data declarations as well as logic control instructions. An instruction is defined to be a line of code or card image. A line of code that contains more that one source statement is still considered one source instruction; a five-line data declaration counts as five instructions. (Boehm 1981)

3.2 Source Code Documentation. Indicate the total number of lines of source code documentation delivered with the software system. The sum of the number of deliverable source instructions in 3.1 and the number of lines of source code documentation equals the total number of lines of code delivered in the software system.

3.3 Deliverable Machine Instructions. Indicate the equivalent number of machine instructions corresponding to the deliverable source instructions in 3.1.

3.4 Database Size. Indicate the total size of the computer databases delivered as part of the defense system software. This refers to the amount of data (in bytes or characters) to be assembled and stored in nonmain storage (that is, tapes, disks, drums, etc.) by the time of system delivery. (Boehm 1981)

4. DIFFICULTY

4.1 Fault Tolerance Requirements. Indicate the types of faults the system must be tolerant to.

4.2 Failure Recovery Mode. Indicate the failure recovery mode.

4.3 Security Requirements. Indicate the level of security required.

4.4 List all organizations developing software which will contribute to the software system. Include other contractors, subcontractors, and government organizations. Also, for each organization (including the reporting contractor), identify all locations where software will be developed (for example, "XYZ Corp. (San Diego, CA; Waltham, MA)").

5. TECHNIQUES EMPLOYED

For each category below, identify the techniques employed at each of the system, subsystem (if any), CPCI, and CPC levels (that is, at least one technique must be identified at each level). Also identify any other levels at which a formal technique is used (for example, module level) and identify the technique. (NASA SEL)

5.1 Specification.

Functional - Components are specified as a set of functions with each component performing a certain action.

Procedural - Components are specified in some algorithmic manner (for example, using a program design language).

English - Components are specified using an English language prose statement of the functions.

Other - Identify any other formal method used to specify the components.

5.2 Design.

Top Down - The implementation of the system one level at a time, beginning at the highest level and expanding downward to subroutines which were yet to be determined at the previous, higher level.

Bottom Up - The implementation of the system starting with the lowest level routines and integrating upward, one level at a time, to the higher level routines.

Iterative Enhancement - The implementation of successive implementations, each producing a usable subset of the final product, until the entire system is fully developed.

Hardest First - The implementation of the most difficult aspects of the system first.

None - No particular strategy has been specified.

Other - Describe the strategy used if it is not a combination of the above.

5.3 Development. Same as 5.2 Design.

5.4 Coding.

Structured Code with Simulated Construct - The programming language does not enforce structured coding techniques (for example, FORTRAN) but structured coding constructs will be simulated.

Structured Code - Structured coding techniques are enforced by the programming language or by some other means, such as preprocessors.

None - No particular coding structure has been specified.

Other - Describe any other coding standards being used.

5.5 Testing.

Top Down - Stubs or dummy procedures are written to handle the yet to be implemented components of the system, and testing begins with the top level routines and proceeds as new components are added at lower levels.

Bottom Up - Checkout of the software is conducted starting with the bottom level modules, using test drivers to simulate the upper level components.

Structure Driven - The structure of the software component is used to determine test cases and test procedures.

Specification Driven - The software specifications are used to determine the test cases and test procedures independent of, or in addition to, the structure of the software.

None - No testing approach has been specified.

Other - Describe any other approach to determining test procedures being used.

5.6 Validation/Verification: Inspection - Visual examination of the design and/or code.

Peer Review - Visual inspection of the code or design by other programmers.

Walk Through - Formal meetings to review the design or code by members of the project team for the purpose of identifying potential problems or improvements.

Proof - Formal proofs of the correctness of the design or code. Describe the techniques used (for example, axiomatic, predicate transforms, functional, etc.).

None - No inspection techniques have been specified.

Other - Describe any other approach to V&V inspection used.

6. FORMALISMS USED

Identify the formalisms used during software development. For each, give the levels (system, subsystem, CPCI, CPC) at which the formalism is being used. (Note: "HOS" refers to an analysis and design methodology developed by Higher Order Software, Inc.) (NASA SEL)

7. AUTOMATED TOOLS USED

Place a check next to the types of automated tools used to support software development. Also, identify any other automated tools used which are not included in the list. Include tools that existed prior to the project and those developed as part of the project. (Boehm 1981)

8. SOFTWARE STANDARDS

List all standards (including in-house standards) that are being applied to the software development. For each standard, give the title, the date of issue, and indicate whether it is required by the contract or optional. (NASA SEL)

9. PROJECT SCHEDULE

9.1 Project Milestones. Give the expected or, if known, the actual date of each listed project milestone. Indicate whether the date given is estimated or actual. If a milestone is being held in increments, indicate the dates of the first and final occurrences and the number of increments. Place "N/A" next to the listed milestones that do not apply to the project. Identify any additional project milestones not listed at which technical progress on the software system development is to be evaluated.

9.2 Perceived Schedule Acceleration/Stretch-out. Indicate the degree of schedule acceleration or stretchout compared to other projects of comparable size and difficulty. Use the following guidelines to determine the ratings:

Severe Acceleration - attempting to complete the development in 75 percent or less of the normal time required for a comparable project without schedule constraints.

Moderate Acceleration - attempting to complete the development in 85 percent of the normal time required for a comparable system without constraints.

Nominal - the schedule is appropriate for a project of this size and complexity.

Moderate Stretchout - the schedule is approximately 130 percent of the time normally required for a comparable project.

Severe Stretchout - the schedule is approximately 160 percent of the time normally required for a comparable project. (Boehm 1981)

10. SYSTEM-LEVEL SOFTWARE-RELATED DOCUMENTATION

For each document listed, give the date the final version is to be delivered and the number of pages in the document that address software (including figures). Indicate whether the date and number of pages are estimated or actual. Identify all other system-level documents developed that relate to software (that is, do not include documents directly associated with individual CPCIs).

11. CORPORATE EXPERIENCE

Indicate the percentage of the software system that falls into each experience category. The percentages are based on the deliverable source instructions defined in 3.1.

PROJECT SUMMARY

PROJECT____________________ DATE__________

CONTRACTOR____________________ CONTRACT NO.__________

1. PROJECT DESCRIPTION

1.1 LIST OF INTERFACING SYSTEMS ____________________

1.2 MAJOR SOFTWARE PRODUCTS ____________________

2. RESOURCES

REUSABLE ITEMS FROM SIMILAR PROJECTS:

PROJECT	# DSI	% DESIGN MOD.	% CODE MOD.	% INTEGR'N REQ'D
		%	%	%
		%	%	%
		%	%	%
		%	%	%
		%	%	%

3. TOTAL SYSTEM SIZE

3.1 DELIVERABLE SOURCE INSTRUCTIONS EXCLUDING SOURCE CODE DOCUMENTATION: __________ INSTRUCTIONS

3.2 LINES OF SOURCE CODE DOCUMENTATION: __________ LINES

3.3 DELIVERABLE MACHINE INSTRUCTIONS: __________ INSTRUCTIONS

3.4 DATABASE SIZE: __________ BYTES

4. DIFFICULTY

4.1 FAULT TOLERANCE REQUIREMENTS: (CHECK THOSE THAT APPLY.)

____ INPUT ERRORS ____ HARDWARE FAILURES ____ SOFTWARE FAILURES ____ NONE

4.2 FAILURE RECOVERY MODE:

____ BATCH ____ REAL TIME

4.3 SECURITY REQUIREMENTS:

____ DEDICATED ____ SYSTEM HIGH ____ MIXED MODE ____ NONE

4.4 LIST OF SOFTWARE DEVELOPMENT ORGANIZATIONS AND SITES: ____________________

Page 9 of 43 pages

5. TECHNIQUES EMPLOYED (IDENTIFY ALL TECHNIQUES USED AT EACH LEVEL)

5.1 SPECIFICATION:

	SYSTEM	SUBSYS	CPCI	CPC	OTHER (SPECIFY)
FUNCTIONAL	____	____	____	____	____________
PROCEDURAL	____	____	____	____	____________
ENGLISH	____	____	____	____	____________
OTHER: ____________	____	____	____	____	____________

5.2 DESIGN:

	SYSTEM	SUBSYS	CPCI	CPC	OTHER (SPECIFY)
TOP DOWN	____	____	____	____	____________
BOTTOM UP	____	____	____	____	____________
ITERATIVE ENHANCE	____	____	____	____	____________
HARDEST FIRST	____	____	____	____	____________
NONE	____	____	____	____	____________
OTHER: ____________	____	____	____	____	____________

5.3 DEVELOPMENT:

	SYSTEM	SUBSYS	CPCI	CPC	OTHER (SPECIFY)
TOP DOWN	____	____	____	____	____________
BOTTOM UP	____	____	____	____	____________
ITERATIVE ENHANCE	____	____	____	____	____________
HARDEST FIRST	____	____	____	____	____________
NONE	____	____	____	____	____________
OTHER: ____________	____	____	____	____	____________

5.4 CODING:

	SYSTEM	SUBSYS	CPCI	CPC	OTHER (SPECIFY)
SIMULATING CONSTRUCT	____	____	____	____	____________
STRUCTURED CODE	____	____	____	____	____________
NONE	____	____	____	____	____________
OTHER: ____________	____	____	____	____	____________

5.5 TESTING:

	SYSTEM	SUBSYS	CPCI	CPC	OTHER (SPECIFY)
TOP DOWN (STUBS)	____	____	____	____	____________
BOTTOM UP (DRIVERS)	____	____	____	____	____________
SPECIFICATION DRIVEN	____	____	____	____	____________
STRUCTURE DRIVEN	____	____	____	____	____________
NONE	____	____	____	____	____________
OTHER: ____________	____	____	____	____	____________

5.6 VALIDATION/VERIFICATION: INSPECTION

	SYSTEM	SUBSYS	CPCI	CPC	OTHER (SPECIFY)
PEER REVIEW	____	____	____	____	____
WALK THROUGHS	____	____	____	____	____
PROOF	____	____	____	____	____
NONE	____	____	____	____	____
OTHER: ____	____	____	____	____	____

6. FORMALISMS USED

	SYSTEM	SUBSYS	CPCI	CPC	OTHER (SPECIFY)
PDL: ____	____	____	____	____	____
HIPO CHARTS	____	____	____	____	____
FLOWCHARTS	____	____	____	____	____
CHAPIN CHARTS	____	____	____	____	____
BASELINE DIAGRAMS (TREE CHARTS)	____	____	____	____	____
HOS	____	____	____	____	____
OTHER: ____	____	____	____	____	____
OTHER: ____	____	____	____	____	____

7. AUTOMATED TOOLS USED

RATING	TOOL	USED
VERY LOW	ASSEMBLER	____
	BASIC LINKER	____
	BASIC MONITOR	____
	BATCH DEBUG AIDS	____
LOW	HOL COMPILER	____
	MACRO ASSEMBLER	____
	SIMPLE OVERLAY LINKER	____
	LANGUAGE INDEPENDENT MONITOR	____
	BASIC SOURCE EDITOR	____
	BASIC LIBRARY AIDS	____
	BASIC DATABASE AIDS	____
NOMINAL	REAL-TIME OR TIME SHARING OPERATING SYSTEM	____
	DATABASE MANAGEMENT SYSTEM	____
	EXTENDED OVERLAY LINKER	____
	INTERACTIVE DEBUG AIDS	____
	SIMPLE PROGRAMMING SUPPORT LIBRARY	____
	INTERACTIVE SOURCE EDITOR	____
HIGH	VIRTUAL MEMORY OPERATING SYSTEM	____
	DATABASE DESIGN AID	____
	SIMPLE PROGRAM DESIGN LANGUAGE	____
	PERFORMANCE MEASUREMENT AND ANALYSIS AIDS	____
	PROGRAMMING SUPPORT LIBRARY WITH BASIC CONFIGURATION MANAGEMENT AIDS	____
	SET-USE STATIC ANALYZER	____
	CONTROL FLOW STATIC ANALYZER	____
	PROGRAM FLOW AND TEST CASE ANALYZER	____
	BASIC TEXT EDITOR AND MANAGER	____
	FILE MANAGER	

7. AUTOMATED TOOLS USED (CONTINUED)

RATING	TOOL	USED
VERY HIGH	FULL PROGRAMMING SUPPORT LIBRARY	____
	DOCUMENTATION SYSTEM	____
	PROJECT CONTROL SYSTEM	____
	REQUIREMENTS SPECIFICATION LANGUAGE AND ANALYZER	____
	EXTENDED DESIGN TOOLS	____
	AUTOMATED VERIFICATION SYSTEM	____
	FAULT REPORT SYSTEM	____
	SPECIAL PURPOSE TOOLS:	
	CROSSCOMPILERS	____
	INSTRUCTION SET SIMULATORS	____
	DISPLAY FORMATTERS	____
	COMMUNICATIONS PROCESSING TOOLS	____
	DATA ENTRY CONTROL TOOLS	____
	CONVERSION AIDS	____
	STRUCTURED LANGUAGE TOOL	____

OTHER AUTOMATED TOOLS: ______________________________

____

____

8. SOFTWARE STANDARDS

TITLE	DATE OF ISSUE	REQ'D	OPT'L

9. PROJECT SCHEDULE

9.1 PROJECT MILESTONES	DATE	EST'D	ACT'L	NUMBER
A. CONTRACT AWARD				
B. SYSTEM REQUIREMENTS REVIEW (SRR)				
C. SYSTEM DESIGN REVIEW (SDR)				
D. PRELIMINARY DESIGN REVIEW (PDR) - FIRST				
E. PDR - FINAL				
F. CRITICAL DESIGN REVIEW (CDR) - FIRST				
G. CDR - FINAL				
H. PRELMINARY QUALIFICATION TEST - FIRST				
I. PQT - FINAL				
J. FORMAL QUALIFICATION TEST (FQT) - FIRST				
K. FQT - FINAL				

PROJECT MILESTONES (CONTINUED)

	DATE	EST'D	ACT'L	NUMBER
L. INTEGRATION OF CPCIS INTO SYSTEM - START				
M. INTEGRATION OF CPCIS INTO SYSTEM - END				
N. DEVELOPMENT TEST & EVALUATION (DT&E) - START				
O. DT&E - END				
P. INITIAL OPERATIONAL TEST & EVALUATION (IOT&E) - START				
Q. IOT&E - END				
R. FUNCTIONAL CONFIGURATION AUDIT (FCA)				
S. PHYSICAL CONFIGURATION AUDIT (PCA)				
T. FORMAL QUALIFICATION REVIEW (FQR)				
U. SYSTEM DELIVERY				
V. CONTRACT END				
OTHER:				
OTHER:				
OTHER:				
OTHER:				

9.2 PERCEIVED SCHEDULE ACCELERATION/STRETCHOUT (CHECK THE APPROPRIATE LEVEL):

A. SEVERE ACCELERATION (75%)	____
B. MODERATE ACCELERATION (85%)	____
C. NOMINAL (100%)	____
D. MODERATE STRETCHOUT (130%)	____
E. SEVERE STRETCHOUT (160%)	____

10. SYSTEM-LEVEL SOFTWARE-RELATED DOCUMENTATION

TITLE	DELIVERY DATE	# PAGES	EST'D	ACT'L
SYSTEM ENGINEERING MANAGEMENT PLAN				
COMPUTER PROGRAM DEVELOPMENT PLAN				
SYSTEM TEST PLAN				
OTHER:				
OTHER:				
OTHER:				
OTHER:				
OTHER:				
OTHER:				

11. CORPORATE EXPERIENCE

GIVE PRECENT OF THE SOFTWARE SYSTEM THAT FALLS INTO EACH CATEGORY (TOTAL = 100%):

A. THE COMPANY HAS BUILT COMPARABLE SYSTEMS WITH SIMILAR REQUIREMENTS. ________%

B. THE COMPANY HAS NEVER BUILT A SYSTEM WITH SIMILAR REQUIREMENTS, BUT SIMILAR SYSTEMS HAVE BEEN BUILT AND TECHNICAL ASSISTANCE IS AVAILABLE TO THE COMPANY. ________%

C. THE COMPANY HAS NEVER BUILT A SYSTEM WITH SIMILAR REQUIREMENTS AND, ALTHOUGH SIMILAR SYSTEMS HAVE BEEN BUILT, NO TECHNICAL ASSISTANCE IS AVAILABLE TO THE COMPANY. ________%

D. PROVED THEORY INDICATES A SYSTEM WITH THESE REQUIREMENTS CAN BE BUILT, BUT A SYSTEM WITH THESE EXACT REQUIREMENTS HAS NEVER BEEN BUILT. ________%

E. THE FEASIBILITY OF BUILDING THE SYSTEM DEPENDS ON THE RESOLUTION OF PIVOTAL RESEARCH PROBLEMS IN THIS TECHNICAL AREA. ________%

PREPARED BY ______________________________ DATE ______________

APPROVED BY ______________________________ DATE ______________

REPORTING MILESTONE ______________________________

INSTRUCTIONS FOR COMPLETING THE CPCI SUMMARY FORM

This form is used to provide information about characteristics of the CPCI that affect its development. A CPCI Summary will be provided for each CPCI according to the schedule specified in the CDRL. Information reported early in the CPCI development will be estimated as accurately as practicable. Intermediate reports will contain actuals if known and updated estimates. The final report will contain all actual data. In cases where the reporting of actual data is limited by measurement accuracy, the reported value should reflect at least 90 percent confidence. Data items shall be continued on separate pages if additional space is needed.

1. DESCRIPTION

1.1 Brief Description. Provide a brief description of the purpose of the CPCI in the system.

1.2 CPCI Functions. List all of the software functions from Table 1 that are performed by the CPCI. Describe any additional functions that are not included in Table 1.

1.3 Development Computer(s). Identify the computer(s) the CPCI is being developed on. (NASA SEL)

1.4 Target Computer(s). Identify the computer(s) the CPCI is targeted for in the system. (NASA SEL)

1.5 Indicate whether any of the target computers are being developed concurrently with the CPCI.

1.6 Virtual Machine Volatility. Indicate the level of virtual machine volatility experienced (or expected) during the CPCI development. The "virtual machine" is the complex of hardware and software that the CPCI calls upon to accomplish its tasks. For example, the virtual machine for an operating system is the computer hardware; the virtual machine for a database oriented user-application system may include the computer hardware, an operating system, and a database management system. Use the following guidelines to determine the rating:

Low - if major changes occur less than every 12 months, or if minor changes occur less than once per month.

Table 1

Software Functions

Type	Category	Index	Function
Operational	Displays	1.1	Avionics
		1.2	Command, Control, & Communications
		1.3	Other
	Avionics	2.1	Mission Planning
		2.2	Navigation
		2.3	Aircraft Steering & Flight Control
		2.4	Sighting, Designation & Location Determination
		2.5	Weapon Delivery
		2.6	Electronic Countermeasures
		2.7	Other
	Command, Control, & Communication	3.1	Network Monitoring
		3.2	Network Control & Switching
		3.3	Sensor Control
		3.4	Signal Processing
		3.5	Message Processing
		3.6	Message Distribution
		3.7	Message Logging & Retrieval
		3.8	Data Reduction
		3.9	Other
	Executive	4.1	Computer Resource Management
		4.2	Computer Operator Interface
		4.3	Other Terminal Operator Interface
		4.4	Special Device Interface
		4.5	Other Input or Output
		4.6	Error Handling/Reconfiguration/Recovery
		4.7	Multicomputer Configuration Control
		4.8	Performance Monitoring & Data Collection
		4.9	Other
	Data Base	5.1	On-Line Data Base Retrieval & Output
		5.2	On-Line Data Base Initialization & Updating
		5.3	Other
	Training	6.1	Control of Exercise Sequencing
		6.2	Operator Performance Data Collection
		6.3	Other
	On-Line Equipment Diagnostic	7.1	System Readiness Test
		7.2	Computer Diagnostic
		7.3	Memory Diagnostic
		7.4	Display Diagnostic
		7.5	Switch/Indicator Panel Diagnostic
		7.6	I/O Diagnostic
		7.7	Mode Diagnostic
		7.8	Other

Table 1

Software Functions (continued)

Type	Category	Index	Function
Support	Operating System	8.1	Computer Resource Management
		8.2	Computer Operator Interface
		8.3	Terminal Operator Interface
		8.4	Input or Output
		8.5	Error Handling/Reconfiguration/Recovery
		8.6	Performance Monitoring & Data Collection
		8.7	Other
	Equipment Maintenance	9.1	Off-Line Computer Diagnostics
		9.2	Other
	Software Development	10.1	Higher-Order Language Compiler
		10.2	Assembler
		10.3	Debugger
		10.4	Loader or Editor
		10.5	Other
	Off-Line Data Base Management	11.1	Data Base Definition
		11.2	Data Base Initialization or Updating
		11.3	Data Base Retrieval & Output Formatting
		11.4	Data Base Restructuring
		11.5	Off-Line Data Base
		11.6	Other
	Design	12.1	Data Base Design
		12.2	Data Base Processor Design
		12.3	Performance Simulation
		12.4	Data Reduction
		12.5	Data Analysis
		12.6	Other
	Test Software	13.1	Test Case Generation
		13.2	Test Case Data Recording
		13.3	Test Data Reduction
		13.4	Test Analysis
		13.5	Other
	Utilities	14.1	Media Conversions
		14.2	Format Translation
		14.3	Sort/Merge
		14.4	Program Library Maintenance
		14.5	Other

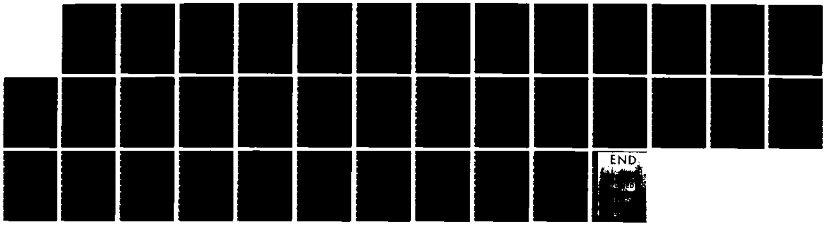
Table 1

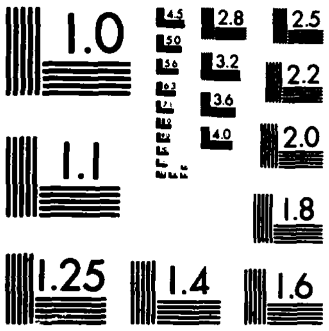
Software Functions (concluded)

Type	Category	Index	Function
Support (cont.)	Off-Line Training	15.1	Data Reduction
		15.2	Training Analysis
		15.3	Scenario Preparation
		15.4	Other
	Project Management	16.1	Project Event Status Accounting
		16.2	Schedule Maintenance/Projection
		16.3	Financial Accounting
		16.4	Software Cost Reporting
		16.5	Hardware Cost Reporting
		16.6	Software Cost Prediction
		16.7	Hardware Cost Prediction
		16.8	Other
	Hardware Subsystem Simulations	17.1	Interfacing Hardware Simulations
		17.2	Environmental Simulations
		17.3	Operator Action Simulations
		17.4	Other

AD-A137 084 SOFTWARE ACQUISITION RESOURCE EXPENDITURE (SARE) DATA COLLECTION METHODOLOGY(U) MITRE CORP BEDFORD MA R L DUMAS DEC 83 MTR-9031 ESD-TR-83-214 2/2

UNCLASSIFIED F19628-82-C-0001 F/G 9/2 NL




1.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Nominal - if major changes occur every 6 months, or minor changes occur every 2 weeks.

High - if major changes occur every 2 months, or minor changes occur every week.

Very High - if major changes occur every 2 weeks, or minor changes occur every 2 days.

A change is considered major if it significantly affects 10 percent or more of the routines under development. A change is considered minor if it affects 1 percent or less of the routines under development. (Boehm 1981)

2. RESOURCES

<u>Reusable Items From Similar Projects</u>. List previous projects that will contribute to the software developed on this project. For each, indicate the approximate number of deliverable source instructions (as defined in 3.1 below) that will be adapted to the current project. Indicate the approximate percentage of the adapted software's design which must be modified to adapt to the new objectives and environment. Indicate the percentage of the adapted software's code which must be modified. Also, indicate the approximate percentage of effort required to integrate and test the adapted software compared to the normal amount of integration and test effort for a new development of comparable size and difficulty. The percentage are of design modification, code modification, and integration may be greater than 100 percent if the effort required is greater than the effort which would have been needed to develop the software from scratch. (Boehm 1981), (NASA SEL)

3. SIZE

3.1 <u>Deliverable Source Instructions</u>. Indicate the total number of deliverable source instructions in the CPCI, excluding source lines that are entirely source code documentation and source instructions from unmodified utility software. Include job control language instructions, format statements, and data declarations as well as logic control instructions. An instruction is defined to be a line of code or card image. A line of code that contains more that one source statement is still considered one source instruction. (Boehm 1981)

3.2 Source Code Documentation. Indicate the total number of lines of source code documentation contained in the CPCI. The sum of the number of deliverable source instructions in 3.1 and the number of lines of source code documentation equals the total number of lines of code delivered in the CPCI.

3.3 Deliverable Machine Instructions. Indicate the equivalent number of machine instructions corresponding to the deliverable source instructions in 3.1.

3.4 Non-Deliverable Support Software. Indicate the approximate number of source instructions developed as support software to aid in the development of the CPCI that are not deliverable as part of the CPCI (for example, test drivers, stubs, debug aids, etc.).

3.5 Database Size. Indicate the size of the computer database(s) accessed by the CPCI. This refers to the amount of data (in bytes or characters) accessed by the CPCI which are to be assembled and stored in nonmain storage (that is, tapes, disks, drums, etc.) by the time of system delivery. (Boehm 1981)

3.6 Size Breakdown By Language. Indicate the percentage of the CPCI source instructions written in each language used. Base the percentages on the number of lines of deliverable source defined in 3.1.

3.7 Size Breakdown By Operation. Indicate the approximate percentage of the deliverable CPCI source instructions which fall into each of the following categories:

Data storage and retrieval - for example, operation of data storage devices, database management, secondary storage handling, data blocking and deblocking.

Online communications - Machine to machine communication with queuing allowed, limited timing restrictions.

Real-time communications and control - machine to machine communications with tight timing constraints, queuing not practical, heavy hardware interface, strict protocol requirements.

Interactive operations - real-time man/machine interfaces, human considerations and error protection very important.

Mathematical operations - routine mathematical operations.

String manipulation - routine applications; typical sorting, formatting, buffer manipulations, etc.

Operating systems - task management, memory management, heavy hardware interfaces, many interactions, high reliability and strict timing requirements.

3.8 Number of Modules. Indicate the number of modules that comprise the CPCI. For the purposes of this data collection form, a module is defined to be the smallest discrete part of a CPCI with an identifiable function and which can be individually compiled or assembled. (NASA SEL)

3.9 Size of Modules. Give the range of module size based on the number of deliverable instructions defined in 3.1. (NASA SEL)

4. SPECIFICATIONS

4.1 Form of Specification. Indicate the form of the CPCI specification and the levels at which each form is used (that is, CPCI, CPC, module, etc.). The specification techniques are defined in paragraph 4.1 of the Instructions for Completing the Project Summary. (NASA SEL)

4.2 Precision of Specification. Rate the precision of the development specification. The specification is very precise if no additional analysis is needed before the detailed design can be developed, precise if only trivial details have to be defined, and imprecise if a great deal of additional analysis is required. (NASA SEL)

5. INTERFACES

5.1 Number of Components Called. Give the number and names of the other CPCIs in the software system that are called by this CPCI. (NASA SEL)

5.2 Number Calling This CPCI. Give the number and names of the other CPCIs in the software system that call this CPCI. (NASA SEL)

5.3 Number of Different Input/Output Formats. Indicate the number of different input formats the CPCI must process and the number of different output formats it produces. Include program calls, interrupts, data base records, displays, and other transactions the CPCI is designed to process or produce.

6. DIFFICULTY

6.1 Percent Utilization. For each of the resources listed, check the range that describes the percentage of the resource that is used during the worst case mode of operation of the CPCI with respect to that resource. The percentage is expressed as the percentage of the total available resource that is used by the CPCI and any other software components concurrently consuming the resource. (Boehm 1981)

6.2 Security. Indicate whether a DoD security classification applies to the CPCI or to any of it inputs or outputs.

6.3 Protection. Indicate whether the CPCI is required to satisfy any privacy or protection requirements.

6.4 Multiple Site Configuration. Indicate the number of sites the CPCI will operate at. In the case of software embedded in a defense item such as an air vehicle, ship, tank, radio, etc., indicate the total number of copies expected to be produced. Also indicate the number of distinct configurations of computers, computer peripherals, and other equipment on which the CPCI will operate.

6.5 Required CPCI Reliability. Indicate the required CPCI reliability using the following guidelines:

Very Low - The effect of a software failure is simply the inconvenience placed on the developers to fix the fault. (For example, demonstration software or an early feasibility-phase simulation model.)

Low - The effect of a software failure is a low level, easily recoverable loss of capability without significant penalty. (For example, off-line training software.)

Nominal - A software failure can result in a moderate loss of system capability, but from which recovery can be achieved without extreme penalty. (For example, off-line equipment diagnostics, utilities, performance monitoring software, etc.)

High - A software failure can result in a major loss of system capability but does not endanger human life. (For example, communications, sensor control, etc.)

Very High - A software failure can result in loss of human life. (For example, aircraft collision avoidance, command and control systems, etc.) (Boehm 1981)

6.6 Complexity. Indicate the level of CPCI complexity. Base the ratings on those given in Table 2 for the various types of functions that may be performed by the CPCI. (Boehm 1981)

7. COMPUTER ACCESS

7.1 Indicate the percentage of source instructions (excluding source code documentation) developed using each type of access to the computer.

7.2 Computer Turnaround Time. Indicate the average turnaround time for a job, that is, the time between when a job is submitted and the time the results are available to the developer. (Boehm 1981)

8. CPCI MILESTONES

Give the expected or, if known, the actual date of each project milestone. Indicate whether the date given is estimated or actual. If a milestone is being held in increments, indicate the dates of the first and final occurances and the number of increments. Place "N/A" next to the listed milestones that do not apply to the project. Identify any additional milestones not listed at which technical progress on the CPCI development is to be evaluated.

9. DOCUMENTATION

For each document listed, give the date the final version is to be delivered and the number of pages in the document (including figures). Indicate whether the dates and number of pages are estimated or actual. Identify all other deliverable and non-deliverable documents developed specifically for the CPCI in addition to those listed.

10. PERSONNEL

10.1 Average Experience of Personnel. Indicate the average experience of the CPCI development personnel in each of the areas listed. (Note: "virtual machine" is defined in 1.6.) (Boehm 1981)

10.2 Average Quality of the CPCI Development Personnel. Rate the average quality of personnel in each category who are involved in the development of the CPCI. The ratings are in terms of percentiles with respect to the overall populations of analysts/designers, programmers, and testers. For example, if the

Table 2

CPCI Complexity Versus Function

Rating	Control Operations	Computational Operations	Device-Dependent Operations	Data Management Operations
Very Low	Straightline code with a few non-nested SP operators: DOs, CASEs, IFTHEN-ELSEs. Simple predicates	Evaluation of simple expressions: for example, A=B+C*(D-E)	Simple read, write statements with simple formats	Simple arrays in main memory
Low	Straight forward nesting of SP operators. Mostly simple predicates	Evaluation of moderate level ex-expressions, e.g., D=SQRT (B**2-4.*A*C)	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level. No cognizance of overlap	Single file subsetting with no data structure changes, no edits, no intermediate files
Nominal	Mostly simple nesting. Some intermodule control. Decision tables	Use of standard math and statistical routines. Basic matrix and vector operations	I/O processing includes device selection, status checking and error processing	Multifile input and single file output. Simple structural changes, simple edits
High	Highly nested SP operators with many compound predicates. Queue and stack	Basic numerical analysis: multivariate interpolation, ordinary differential equations.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc). Optimized I/O overlap	Special purpose subroutines activated by data stream contents. Complex data restructuring at record level

Table 2 (Concluded)

Rating	Control Operations	Computational Operations	Device-Dependent Operations	Data Management Operations
	control. considerable intermodule control	Basic truncation, round-off concerns		
Very high	Reentrant and recursive coding. Fixed-priority interrupt handling	Difficult but structured NA: nearsingular matrix equations, partial differential equations	Routines for interrupt diagnosis, servicing, masking. Communication line handling	A generalized, parameter-driven file structuring routing. File building, command processing, search optimization
Extra high	Multiple resource scheduling with dynamically changing priorities. Microcode-level control	Difficult and unstructured NA: highly accurate analysis of noisy, stochastic data	Device timing-dependant coding, microprogrammed operations	Highly coupled dynamic relational structures. Natural language data management

analysts/designers are considered average (50 percent) check the box under 36 to 55 percent. The ratings should take into consideration basic ability, efficiency and thoroughness, and ability to communicate and work with others. The ratings should not consider experience, which is covered in 10.1. (Boehm 1981)

10.3 Experience With Modern Programming Practices. Indicate the relative experience of the CPCI development personnel using modern programming practices (defined in 5. of the Project Summary Form). Use the following guidelines to determine the ratings:

Very Low - no use of modern programming practices.

Low - Beginning, experimental use of some modern programming practices.

Nominal - Reasonably experienced in the use of some modern programming practices.

High - Reasonably experienced in the use of most modern programming practices.

Very High - Routine use of all modern programming practices. (Boehm 1981)

10.4 Indicate how the experience levels and quality of the CPCI development team were determined.

11. SOFTWARE CHANGES

Indicate the number of cost and no cost engineering change proposals (ECPs) submitted during each phase of the CPCI development that impacted the CPCI. Indicate the number of ECPs approved during each phase and the sum of the costs of the approved ECPs. Also indicate the number of software trouble reports opened and closed during each phase of the CPCI development.

CPCI SUMMARY

PROJECT ____________________ DATE __________

CPCI ____________________ CONFIGURATION NO. __________

1. DESCRIPTION

1.1 BRIEF DESCRIPTION ____________________

1.2 CPCI FUNCTIONS - LIST ALL FUNCTIONS FROM TABLE 1 THAT ARE PERFORMED BY THE CPCI:

TYPE	CATEGORY	INDEX	FUNCTION
OTHER:			
OTHER:			

1.3 DEVELOPMENT COMPUTER(S) ____________________

1.4 TARGET COMPUTER(S) ____________________

1.5 IS THE TARGET COMPUTER BEING DEVELOPED CONCURRENTLY WITH THE CPCI? __________

1.6 VIRTUAL MACHINE VOLATILITY (CHECK THE APPROPRIATE LEVEL):

A. LOW ____

B. NOMINAL ____

C. HIGH ____

D. VERY HIGH ____

2. RESOURCES

REUSABLE ITEMS FROM SIMILAR PROJECTS:

PROJECT/COMPONENT	# DSI	% DESIGN MOD.	% CODE MOD.	% INTEGR'N REQ'D
		%	%	%
		%	%	%
		%	%	%
		%	%	%
		%	%	%

3. SIZE

3.1 DELIVERABLE SOURCE INSTRUCTIONS EXCLUDING SOURCE CODE DOCUMENTATION: ________ INSTRUCTIONS

3.2 LINES OF SOURCE CODE DOCUMENTATION: ________ LINES

3.3 DELIVERABLE MACHINE INSTRUCTIONS: ________ INSTRUCTIONS

3.4 NON-DELIVERABLE SUPPORT SOFWARE: ________ INSTRUCTIONS

3.5 DATABASE SIZE: ________ BYTES

3.6 SIZE BREAKDOWN BY LANGUAGE (TOTAL = 100%):

LANGUAGE	PERCENTAGE	LANGUAGE	PERCENTAGE
ASSEMBLY	____%	ALGOL	____%
COBOL	____%	FORTRAN	____%
JOVIAL	____%	PL/I	____%
ADA	____%	MICROCODE	____%
OTHER: ________	____%	OTHER: ________	____%
OTHER: ________	____%	OTHER: ________	____%

3.7 SIZE BREAKDOWN BY OPERATION (TOTAL = 100%):

A. DATA STORAGE AND RETRIEVAL ________%

B. ONLINE COMMUNICATIONS ________%

C. REAL-TIME COMMAND AND CONTROL ________%

D. INTERACTIVE OPERATIONS ________%

E. MATHEMATICAL OPERATIONS ________%

F. STRING MANIPULATION ________%

G. OPERATING SYSTEMS ________%

3.8 NUMBER OF MODULES: ________

3.9 SIZE OF MODULES: SMALLEST ________ LARGEST ________ AVERAGE ________

4. SPECIFICATIONS

4.1 FORM OF SPECIFICATION: (CHECK ALL THAT ARE USED AND GIVE THE LEVEL)

	CPCI	CPC	OTHER (SPECIFY)
A. FUNCTIONAL	____	____	________
B. PROCEDURAL	____	____	________
C. ENGLISH	____	____	________
D. OTHER: ________	____	____	________

4.2 PRECISION OF SPECIFICATION:

A. VERY PRECISE ____ B. PRECISE ____ C. IMPRECISE ____

5. INTERFACES

5.1 NUMBER OF COMPONENTS CALLED: ________ NAMES: ______________________________

5.2 NUMBER CALLING THIS CPCI: ________ NAMES: ______________________________

5.3 NUMBER OF DIFFERENT I/O FORMATS: INPUT ____________ OUTPUT ____________

6. DIFFICULTY

6.1 PERCENT UTILIZATION:	$\leq$ 50%	51% TO 70%	71% TO 85%	86% TO 95%	>95%
A. MAIN STORAGE					
B. PERIPHERAL STORAGE					
C. EXECUTION TIME					

6.2 SECURITY: DOES A DOD SECURITY CLASSIFICATION APPLY TO THE CPCI OR ANY OF ITS INPUTS/OUTPUTS? ________

6.3 PROTECTION: IS THE CPCI REQUIRED TO SATISFY ANY PRIVACY OR PROTECTION REQUIREMENTS? ________

6.4 MULTIPLE SITE CONFIGURATION:

A. NUMBER OF DISTINCT SITES ____________

B. NUMBER OF DISTICT CONFIGURATIONS ____________

6.5 REQUIRED CPCI RELIABILITY (CHECK APPROPRIATE LEVEL):

A. VERY LOW ____

B. LOW ____

C. NOMINAL ____

D. HIGH ____

E. VERY HIGH ____

6.6 COMPLEXITY (CHECK THE APPROPRIATE LEVEL):

A. VERY LOW ____

B. LOW ____

C. NOMINAL ____

D. HIGH ____

E. VERY HIGH ____

7. COMPUTER ACCESS

7.1 PERCENTAGE OF SOURCE INSTRUCTIONS DEVELOPED USING EACH OF THE FOLLOWING (TOTAL = 100%):

A. BATCH	______ %
B. DEDICATED PROCESSOR	______ %
C. TEST BED WITH HIGH PRIORITY	______ %
D. TEST BED WITH LOW PRIORITY	______ %
E. INTERACTIVE	______ %

7.2 COMPUTER TURNAROUND TIME:

A. LOW (INTERACTIVE)	____
B. NOMINAL (< 4 HRS)	____
C. HIGH (4 TO 12 HRS)	____
D. VERY HIGH (> 12 HRS)	____

8. CPCI MILESTONES

MILESTONES	DATE	EST'D	ACT'L	NUMBER
A. DESIGN START				
B. PRELIMINARY DESIGN REVIEW (PDR) - FIRST				
C. PDR - FINAL				
D. DEVELOPMENT SPECIFICATION APPROVAL				
E. CRITICAL DESIGN REVIEW (CDR) - FIRST				
F. CDR - FINAL				
G. CODING & DEBUG - START				
H. CODING & DEBUG - COMPLETION				
I. INFORMAL TEST AND INTEGRATION - START				
J. INFORMAL TEST AND INTEGRATION - COMPLETION				
K. PRELIMINARY QUALIFICATION TEST (PQT) - FIRST				
L. PQT - FINAL				
M. FORMAL QUALIFICATION TEST (FQT) - FIRST				
N. FQT - FINAL				
O. PRODUCT SPECIFICATION APPROVAL				
P. FUNCTIONAL CONFIGURATION AUDIT (FCA)				
Q. PHYSICAL CONFIGURATION AUDIT (PCA)				
OTHER:				
OTHER:				
OTHER:				

9. DOCUMENTATION

TITLE	DELIVERY DATE	# PAGES EST'D	ACT'L
A. CPCI DEVELOPMENT SPECIFICATION	______	______	______
B. CPCI PRODUCT SPECIFICATION	______	______	______
C. TEST PLAN	______	______	______
D. TEST PROCEDURES	______	______	______
E. TEST REPORT	______	______	______
F. USER'S MANUAL	______	______	______
G. OTHER: ______	______	______	______
H. OTHER: ______	______	______	______
I. OTHER: ______	______	______	______
J. OTHER: ______	______	______	______

10. PERSONNEL

10.1 AVERAGE EXPERIENCE OF PERSONNEL

	≤ 4 MOS	4 MOS TO 1 YR	1 TO 3 YRS	3 TO 6 YRS	≥ 6 YEARS
A. APPLICATION AREA	____	____	____	____	____
B. TECHNIQUES TO BE USED	____	____	____	____	____
C. LANGUAGES TO BE USED	____	____	____	____	____
D. VIRTUAL MACHINE	____	____	____	____	____
E. SUPPORT SOFTWARE/TOOLS	____	____	____	____	____

10.2 AVERAGE QUALITY OF THE CPCI DEVELOPMENT PERSONNEL (PERCENTILES):

	≤ 15%	16 - 35%	36 - 55%	56 - 75%	76 - 90%	≥ 90%
A. ANALYSTS/DESIGNERS	____	____	____	____	____	____
B. PROGRAMMERS	____	____	____	____	____	____
C. TESTERS	____	____	____	____	____	____
D. OVERALL	____	____	____	____	____	____

10.3 EXPERIENCE WITH MODERN PROGRAMMING PRACTICES:

A. VERY LOW	____
B. LOW	____
C. NOMINAL	____
D. HIGH	____
E. VERY HIGH	____

10.4 PERSONNEL EVALUATION IS BASED ON:

A. CORPORATE AVERAGES _____

B. SPECIFIC TEAM MEMBERS _____

C. OTHER: ____

11. SOFTWARE CHANGES

	ENGINEERING CHANGE PROPOSALS			S/W TROUBLE REPORTS	
PHASE	# SUBMITTED	# APPROVED	EST. COST	OPENED	CLOSED
A. PRELIMINARY DESIGN (CONTRACT AWARD TO PDR)	_____	_____	$_____	_____	_____
B. DETAILED DESIGN (PDR TO CDR)	_____	_____	$_____	_____	_____
C. CODE & DEBUG (CDR TO T&I START)	_____	_____	$_____	_____	_____
D. TEST & INTEGRATION (T&I START TO FQT)	_____	_____	$_____	_____	_____
E. SYSTEM-TEST/IOC (FQT TO CONTRACT END)	_____	_____	$_____	_____	_____
TOTALS	_____	_____	$_____	_____	_____

PREPARED BY ______________________________ DATE _____________

APPROVED BY ______________________________ DATE _____________

REPORTING MILESTONE __________________________

INSTRUCTIONS FOR COMPLETING THE CPC SUMMARY FORM

The CPC Summary is used to provide information about characteristics of the CPC that affect its development. A CPC Summary will be provided for each CPC according to the schedule specified in the CDRL. Information reported early in the CPC development will be estimated as accurately as practicable. Intermediate reports will contain actuals if known and updated estimates. The final report will contain all actual data. In cases where the reporting of actual data is limited by measurement accuracy, the reported value should reflect at least 90 percent confidence. Data items shall be continued on separate pages if additional space is needed.

1. DESCRIPTION

1.1 Brief Description. Provide a brief description of the of the purpose of the CPC within the CPCI.

1.2 Software Functions. List all of the software functions from Table 1 that are performed by the CPC. Table 1 is provided in the Instructions for Completing the CPCI Summary Form. Describe any additional functions that are not included in Table 1.

1.3 Development Computer(s). Identify the computer(s) the CPC is being developed on. (NASA SEL)

1.4 Target Computer(s). Identify the computer(s) the CPC is targeted for in the system. (NASA SEL)

2. SIZE

2.1 Deliverable Source Instructions. Indicate the total number of deliverable source instructions in the CPC, excluding source lines that are entirely source code documentation and source instructions from unmodified utility software. Include job control language instructions, format statements, and data declarations as well as logic control instructions. An instruction is defined to be a line of code or card image. A line of code that contains more that one source statement is still considered one source instruction. (Boehm 1981)

2.2 Source Code Documentation. Indicate the total number of lines of source code documentation contained in the CPC. The sum of the number of deliverable source instructions in 2.1 and the number of lines of source code documentation equals the number of lines of code delivered in the CPC.

2.3 Deliverable Machine Instructions. Indicate the equivalent number of machine instructions corresponding to the deliverable source instructions in 2.1.

2.4 Non-Deliverable Support Software. Indicate the approximate number of source instructions developed as support software to aid in the development of the CPC that are not deliverable as part of the CPC (for example, test drivers, stubs, debug aids, etc.).

2.5 Database Size. Indicate the size of the database(s) accessed by the CPC. This refers to the amount of data (in bytes or characters) accessed by the CPC that is to be assembled and stored in nonmain storage (that is, tapes, disks, drums, etc.) by the time of system delivery. (Boehm 1981)

2.6 Size Breakdown By Language. Indicate the percentage of deliverable source instructions defined in 2.1 which are written in each language.

2.7 Size Breakdown By Operation. Indicate the approximate percentage of deliverable source instructions defined in 2.1 which fall into each category. The categories are defined in 3.7 of the CPCI Summary Form.

2.8 Number of Modules. Indicate the number of modules that comprise the CPC. For the purposes of this data collection form, a module is defined to be the smallest discrete part of a CPC with an identifiable function and which can be individually compiled or assembled. (NASA SEL)

2.9 Size of Modules. Give the range of module size based on the number of deliverable source instructions defined in 2.1. (NASA SEL)

3. INTERFACES

3.1 Number of Other CPCs Called. Give the number and names of the other CPCs in the CPCI that are called by this CPC. (NASA SEL)

3.2 Number Calling This CPC. Give the number and names of the other CPCs in the CPCI that call this CPC. (NASA SEL)

3.3 Number of Different Input/Output Formats. Indicate the number of different input formats the CPC must process and the number of different output formats it produces. Include program calls, interrupts, data base records, displays, and other transactions the CPC is designed to process or produce.

4. UTILIZATION

For each of the resources listed, check the range that corresponds to the percentage of the resource that is used during the worst case mode of operation for the CPC with respect to that resource. The percentage is expressed as the percentage of the total available resource that is used by the CPC and all other software components concurrently consuming the resource. (Boehm 1981)

CPC SUMMARY

PROJECT ______________________________ DATE ______________

CPC ______________________________ CONFIGURATION NO. ______________

1. DESCRIPTION

1.1 BRIEF DESCRIPTION ______________________________

____

1.2 SOFTWARE FUNCTIONS (LIST ALL FUNCTIONS FROM TABLE 1 THAT APPLY)

TYPE	CATEGORY	INDEX	FUNCTION
OTHER:			
OTHER:			

1.3 DEVELOPMENT COMPUTER(S) ______________________________

1.4 TARGET COMPUTER(S) ______________________________

2. SIZE

2.1 DELIVERABLE SOURCE INSTRUCTIONS EXCLUDING SOURCE CODE DOCUMENTATION: ____________ INSTRUCTIONS

2.2 LINES OF SOURCE CODE DOCUMENTATION: ____________ LINES

2.3 DELIVERABLE MACHINE INSTRUCTIONS: ____________ INSTRUCTIONS

2.4 NON-DELIVERABLE SUPPORT SOFWARE: ____________ INSTRUCTIONS

2.5 DATABASE SIZE: ____________ BYTES

2.6 SIZE BREAKDOWN BY LANGUAGE (TOTAL = 100%):

LANGUAGE	PERCENTAGE	LANGUAGE	PERCENTAGE
ASSEMBLY	%	ALGOL	%
COBOL	%	FORTRAN	%
JOVIAL	%	PL/I	%
ADA	%	MICROCODE	%
OTHER:	%	OTHER:	%
OTHER:	%	OTHER:	%

2.7 SIZE BREAKDOWN BY OPERATION (TOTAL = 100%):

A. DATA STORAGE AND RETRIEVAL	______%
B. ONLINE COMMUNICATIONS	______%
C. REAL-TIME COMMAND AND CONTROL	______%
D. INTERACTIVE OPERATIONS	______%
E. MATHEMATICAL OPERATIONS	______%
F. STRING MANIPULATION	______%
G. OPERATING SYSTEMS	______%

2.8 NUMBER OF MODULES: ______

2.9 SIZE OF MODULES: SMALLEST ______ LARGEST ______ AVERAGE ______

3. INTERFACES

3.1 NUMBER OF OTHER CPCS CALLED: ______ NAMES: ______

3.2 NUMBER CALLING THIS CPC: ______ NAMES: ______

3.3 NUMBER OF DIFFERENT I/O FORMATS: INPUT ______ OUTPUT ______

4. UTILIZATION

PERCENT UTILIZATION:	≤ 50%	51% TO 70%	71% TO 85%	86% TO 95%	≥ 95%
A. MAIN STORAGE	___	___	___	___	___
B. PERIPHERAL STORAGE	___	___	___	___	___
C. EXECUTION TIME	___	___	___	___	___

PREPARED BY ______ DATE ______

APPROVED BY ______ DATE ______

REPORTING MILESTONE ______

INSTRUCTIONS FOR COMPLETING THE DATABASE SUMMARY FORM

A Database Summary will be provided for each computer database to be assembled and stored on nonmain storage (that is, disks, drums, tapes, etc.) and delivered under the contract. This includes databases dedicated to individual CPCIs as well as databases accessed by multiple CPCIs. Information reported early in the project will be estimated as accurately as practicable. Intermediate reports will contain actuals if known and updated estimates. The final report will contain all actual data. In cases where the reporting of actual data is limited by measurement accuracy, the reported value should reflect at least 90 percent confidence.

1. SIZE

1.1 Database Size. Indicate the size of the database in bytes. Database size refers to the amount of data (in bytes or characters) to be assembled and stored on nonmain storage (that is, disks, drums, tapes, etc.) by the time of system delivery. (Boehm 1981)

1.2 Byte Size. Indicate the number of bits per byte.

2. SOFTWARE COMPONENTS ACCESSING THE DATABASE

Identify the CPCIs that access the database. Indicate the percentage of the database accessed by each CPCI and indicate whether the percentage is estimated or actual.

DATABASE SUMMARY

PROJECT ______________________________ DATE ______________

DATABASE ______________________________ CONFIGURATION NO. ______________

BRIEF DESCRIPTION ______________________________

____

1. SIZE

1.1 DATABASE SIZE: ______________ BYTES

1.2 BYTE SIZE: ______________ BITS

2. SOFTWARE COMPONENTS ACCESSING THE DATABASE

CPCI	PERCENT	EST'D	ACT'L
	%		
	%		
	%		
	%		
	%		
	%		
	%		
	%		
	%		
	%		

PREPARED BY ______________________________ DATE ______________

APPROVED BY ______________________________ DATE ______________

REPORTING MILESTONE ______________________________

INSTRUCTIONS FOR COMPLETING THE RESOURCE EXPENDITURE SUMMARY FORM

This form provides information on resources expended under software-related work breakdown structure elements. The reporting schedule and the WBS reporting levels are specified in the CDRL.

FORM A

1. PROJECT COST SUMMARY

1.1 Negotiated Cost. Enter the total contract cost (excluding fee or profit) on which agreement has been reached as of the cutoff date of the report. For an incentive contract, enter the definitized contract target cost. Amounts for changes will not be included in this item until they have been priced and included in the contract through contractual change order or supplemental agreement. For a fixed-fee contract, enter the estimated cost negotiated. Changes to the estimated cost will consist only of amounts for changes in the contract scope of work, not for cost growth.

1.2 Target Price. Enter the negotiated cost plus profit/fee applicable to the definitized contractual effort.

1.3 Estimated Cost of Authorized, Unpriced Work. Enter the estimated dollar amount (excluding fee or profit) for all contract changes for which written authorization has been received but for which definitized prices have not been incorporated in the contract through supplemental agreement.

1.4 Estimated Price. Enter the latest revised estimate of the final price of the contract to the government, including the cost of all authorized contractual work and applicable profit/fee, incentives, and cost sharing provisions.

2. OVERHEAD, G&A AND FEES

2.1 Overhead Rate. Enter the overhead rate as a percentage of direct costs.

2.2 General and Administrative. Enter the General and Administrative (G&A) costs.

2.3 Profit/Fee. Enter the fee or profit percentage which will apply if the negotiated cost of the contract is met.

2.4 Share Ratio(s). Enter the cost sharing ratio(s) applicable to costs over/under the negotiated contract costs.

3. DIRECT LABOR CHARGES

3.1 Check all types of labor that are included in direct labor hours charged to WBS elements.

3.2 Indicate whether or not direct labor hours reported on Form B include uncompensated overtime. If not, estimate the percentage of uncompensated overtime as a function of direct labor hours.

FORM B

Month. Indicate the month and year corresponding to the column.

WBS Element. Give the indexes and names of the WBS elements.

Man Hours. Indicate the number of direct labor hours expended under the WBS element during the month. The number of hours should include overtime, even if the employees are exempt from overtime compensation. It should exclude overhead hours such as sick time, vacation, and personal time.

Labor Dollars. Indicate the direct labor cost for the WBS element during the month.

Total Dollars. Indicate the total cost for the WBS element for the month.

RESOURCE EXPENDITURE SUMMARY - FORM A

PROJECT ______________________ DATE ____________

CONTRACTOR ______________________ CONTRACT NO. ____________

1. PROJECT COST SUMMARY

1.1	NEGOTIATED COST	$________
1.2	TARGET PRICE	$________
1.3	ESTIMATED COST OF AUTHORIZED, UNPRICED WORK	$________
1.4	ESTIMATED PRICE	$________

2. OVERHEAD, G&A AND FEES

2.1	OVERHEAD RATE	________%
2.2	GENERAL & ADMINISTRATIVE	$________
2.3	PROFIT/FEE	________%
2.4	SHARE RATIO(S)	________

3. DIRECT LABOR CHARGES

3.1 INDICATE ALL TYPES OF LABOR INCLUDED IN DIRECT LABOR HOURS:

______	TECHNICAL STAFF (ENGINEERS, PROGRAMMERS)	______	HIGHER MANAGEMENT (PROGRAM MANAGERS)
______	TECHNICAL PROJECT MANAGERS	______	SECRETARIES
______	PROGRAM LIBRARIANS	______	TECHNICAL AIDES
______	OTHER: ____________	______	OTHER: ____________

3.2 DO THE LABOR HOURS REPORTED ON FORM B INCLUDE UNCOMPENSATED OVERTIME? ________

IF NOT, ENTER ESTIMATED PERCENT OF UNCOMPENSATED OVERTIME: ________%

PREPARED BY ______________________ DATE ____________

APPROVED BY ______________________ DATE ____________

REPORTING MILESTONE ______________________

RESOURCE EXPENDITURE SUMMARY - FORM B

PROJECT ______________________ DATE ______________

CONTRACTOR ______________________

WBS ELEMENT	MONTH ________			MONTH ________			MONTH ________		
	MAN HRS	LABOR $	TOTAL $	MAN HRS	LABOR $	TOTAL $	MAN HRS	LABOR $	TOTAL $

(DOLLAR AMOUNTS IN THOUSANDS OF DOLLARS)

ATTACHMENT 3

SARE EVALUATION QUESTIONNAIRE

<u>Instructions</u>

This questionnaire is intended to help reviewers express opinions about the proposed SARE documents. However, it is not intended to limit the flow of ideas. Please feel free to provide additional comments outside the structure of this questionnaire, including returning "red-lined" copies of the documents themselves.

Please answer the questions frankly and honestly. We will accept your views as an individual and/or as the official position of your organization.

Your contribution to this effort is greatly appreciated. Please return this questionnaire by 1 April 1984 to:

Headquarters, Electronic Systems Division
Director of Cost Analysis
Management and Information Systems Division
Hanscom Air Force Base, MA 01731
Attention: Capt. J. P. Dean, ESD/ACCE

Name______________________________ Date______________

Organization____

Address___________________________ Telephone__________

____

____

SARE EVALUATION QUESTIONNAIRE

PART I - BACKGROUND

1. Are you responding for yourself or your organization?

   _____ yourself
   _____ your organization

2. Involvement in software cost estimation.

   a. Does your organization prepare software cost estimates? ________

   If yes, for what purpose? (check all that apply)

   ____ bidding contracts
   ____ internal management
   ____ source selection
   ____ monitoring (sub)contractor costs

   ____ other:____

   b. Does your organization use software cost estimates? ________

   If yes, for what purpose? (check all that apply)

   ____ internal management
   ____ source selection
   ____ monitoring (sub)contractor costs

   ____ other:____

   c. Does your organization perform software cost estimation research? ________

   If yes, which of the following? (check all that apply)

   ____ develop/maintain in-house software cost models
   ____ evaluate/calibrate commercial models
   ____ develop software cost databases

   ____ other:____

3. Involvement in software acquisition.

What is your organization's involvement in software acquisition? (check all that apply)

____ buy/contract for software
____ develop/supply software products
____ establish government policy with regard to software acquisition

____ other: ________________________________

4. Status of software management in your organization.

a. Do you have a standard work breakdown structure (WBS) for software? ________

If yes, to whom is it applied? (check all that apply)

____ internally
____ (sub)contractors

____ other: ________________________________

b. Do you have standard data collection on software projects? ________

If yes, for what purpose? (check all that apply)

____ internal management
____ monitor (sub)contractor costs
____ create a database to support software cost estimation

____ other: ________________________________

c. Do you use automated aids to support software cost estimation or software cost management? ________

If yes, what types? (check all that apply)

____ in-house developed cost model(s)
____ commercial software cost models. Which? (optional)

____

____

____

____ other aids (for example, cost accounting system). Which?

____

____

____

5. Have you ever used the following on a project?

   a. MIL-STD-881A, "Work Breakdown Structures for Defense Materiel Items?" ________

   b. DoDI 7000.2, "Performance Measurement for Selected Acquisitions" (which includes the cost/schedule control systems criteria)? ________

6. What types of software systems do you deal with?

____ Weapon System	____ Avionics
____ C3	____ Business/Financial
____ Other:______________________________	

7. What size software systems do you deal with? (Check all that apply).

   ____ less than 50K lines of code

   ____ 50K to 100K lines of code

   ____ 100K to 250K lines of code

   ____ Over 250K lines of code

SARE EVALUATION QUESTIONNAIRE

PART II - PROPOSED SOFTWARE WORK BREAKDOWN STRUCTURE

1. What is your overall evaluation of the proposed WBS?

____

____

____

____

2. a. Are the government and contractor responsibilities clear? _______

If not, what is not clear?

____

____

____

b. In your opinion, are the government and contractor responsibilities complete and appropriate? _______

If not, what is missing or inappropriate?

____

____

____

c. In your opinion, are the WBS requirements and definitions consistent with existing military regulations and standards? _______

If not, what is not correct?

____

____

____

d. Is the proposed software WBS appropriate for the types of systems and software your organization deals with? ________

If not, what is needed to make it appropriate?

____

____

____

e. Is the WBS at an appropriate level of detail? ________

If not, what level would you suggest?

____

____

____

3. Can you recommend any improvements (additions, modifications, or deletions) to the document?

____

____

____

____

____

4. Do you feel the document should become a new military standard, be incorporated into a revision of MIL-STD-881A, be used as guidance only, or not be pursued at all?

_____ New military standard

_____ Revision of MIL-STD-881A

_____ Guidance only

_____ Not pursued at all

5. If your organization has a standard WBS for software, we would like to see it. If possible, please enclose a copy with this questionnaire.

_____ Enclosed is a copy of our software WBS.

_____ Our software WBS is proprietary. (Can arrangements be made to see it? ________)

_____ We do not have a standard WBS for software.

SARE EVALUATION QUESTIONNAIRE

PART III - PROPOSED DATA ITEM DESCRIPTION

1. What is your overall evaluation of the data item description?

____

____

____

____

2. a. Are the preparation instructions clear? ______

If not, what is not clear?

____

____

____

b. Are the preparation instructions complete? ______

If not, what is missing?

____

____

____

c. Are the definitions of the data items clear (unambiguous)? ______

If not, which are not clear?

____

____

____

d. Are the data items complete (that is, do they cover all major factors that influence cost)? ________

If not, what is missing?

____

____

____

e. Are the data items defined appropriately? ________

If not, which are not appropriate?

____

____

____

f. Can the software cost models used by your organization be calibrated using the data items? ________

If not, what is missing?

____

____

____

3. Can you recommend any improvements (additions, modifications, or deletions) to the document?

____

____

____

____

4. Do you feel the document should become an official data item description? ________

If not, why not?

____

____

____

5. If your organization collects data on software projects to support cost estimation, we would like to know what kinds of data you collect. If possible, please enclose a copy of your data collection forms and definitions.

_____ Enclosed is a copy of our data collection forms and definitions.

_____ Our data collection forms are proprietary. (Can arrangements be made to see them? ________)

_____ We do not collect that kind of data on software projects.

SARE EVALUATION QUESTIONNAIRE

PART IV - SUMMARY

1. a. How much would you estimate data collection of this kind would add to the cost of a defense system acquisition program (as a percentage of total software cost)?

_____ Less than 2 percent

_____ 2 percent to 5 percent

_____ 5 percent to 10 percent

_____ Greater than 10 percent

b. In your judgement, would such added cost be worth it? ________

If not, please explain.

____

____

____

2. Any further comments?

____

____

____

____

____